MÉTODOS DE BRANCH-AND-BOUND E PENALIDADES
EM PROGRAMAÇÃO LINEAR EM DOIS NÍVEIS

Carlos Henrique Medeiros de Sabóia

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profa. Susana Scheimberg de Makler, D.Sc.

Prof. Manoel Bezerra Campêlo Neto, D.Sc.

Prof. José Herskovits Norman, Dr.Ing.

Profa. Ana Friedlander de Martínez Perez, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2002

MEDEIROS DE SABÓIA, CARLOS HENRIQUE

Métodos de branch-and-bound e penalidades em programação linear em dois níveis [Rio de Janeiro] 2002

VIII, 85 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Programação Linear em Dois Níveis.

2. Otimização Global.

3. Métodos de Penalidade e Branch-and-Bound.

I. COPPE/UFRJ II. Título (série).

Aos meus queridos pais

e irmãos.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MÉTODOS DE BRANCH-AND-BOUND E PENALIDADES
EM PROGRAMAÇÃO LINEAR EM DOIS NÍVEIS

Carlos Henrique Medeiros de Sabóia

Junho/2002

Orientadores: Susana Scheimberg de Makler
Manoel Bezerra Campêlo Neto

Programa: Engenharia de Sistemas e Computação

Este trabalho trata do problema de programação linear em dois níveis. Suas características e propriedades básicas são apresentadas, bem como os principais algoritmos existentes na literatura voltados a sua resolução.

Um estudo comparativo entre dois algoritmos globais conhecidos na literatura é realizado. O primeiro algoritmo faz uso de um método de penalidade associado a um algoritmo do tipo *outer approximation*. O segundo algoritmo baseia-se em um método *branch-and-bound*.

Foram propostos neste trabalho os algoritmos de penalidade e *branch-and-bound* modificados. Os mesmos contornaram as deficiências apresentadas pelos algoritmos originais nos testes realizados. As versões modificadas foram desenvolvidas através da introdução de um método enumerativo, no algoritmo de penalidade original, e de um método de penalidade, no algoritmo *branch-and-bound* original.

Estes novos algoritmos mostram ser bem mais eficientes, do ponto de vista computacional, que os originais.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

BRANCH-AND-BOUND AND PENALTY METHODS IN
LINEAR BILEVEL PROGRAMMING

Carlos Henrique Medeiros de Sabóia

June/2002

Advisors: Susana Scheimberg de Makler
Manoel Bezerra Campêlo Neto

Department: Computing and Systems Engineering

This work deals with the linear bilevel programming problem. Its characteristics, properties and the main global algorithms known in literature are presented.

A computational study of two global algorithms is carried out. The first algorithm uses a penalty method together with an outer approximation procedure. The second one is based on a branch-and-bound approach.

In this work penalty and branch-and-bound modified algorithms were proposed. They overcome the trouble presented by the original algorithms which were observed in the computational tests. The modified versions were developed by the introduction of an enumerative method in the original penalty algorithm, and by the inserting of a penalty method in the original branch-and-bound algorithm.

The obtained algorithms present a better performance, from the computational point of view, than the original ones.

Agradecimentos

Agradeço acima de tudo à DEUS.

Agradeço aos meus pais Helder e Gilka por TUDO, aos meus irmãos Ticiana e Marquinhos, aos meus avós, tios e tias.

Aos professores do curso de graduação sou grato ao Prof. Felipe Loureiro do Departamento de Engenharia de Transportes da UFC pelas oportunidades oferecidas e pelo incentivo ao curso de mestrado e em especial expresso a minha gratidão ao Prof. Antônio Clécio Fontelles Thomaz do Departamento de Estatística e Matemática Aplicada da UFC pela sua amizade, confiança e pelo apoio dado para o ingresso no mestrado.

Aos meus orientadores do curso de mestrado sou extremamente grato e expresso o meu respeito a Profa. Susana Scheimberg, pela sua amizade, confiança em meu trabalho, pelos seus conselhos e por sua extrema competência no ensino e pesquisa. Ao professor e amigo Manoel Campêlo, orientador da UFC, sou grato pela sua amizade, atenção e por suas valiosas contribuições ao desenvolvimento deste trabalho.

Aos vários amigos e amigas que fiz ao longo do curso. Em especial aos, para sempre, amigos que moraram na república em Ipanema, Emílio César, João Quariguazi, Beethowen Nepomuceno, Marley Júnior, Sávio Freire, Wilson e Daniel. Aos meus amigos da COPPE/Sistemas, Jorge, Sérgio, Tiberius (T), André, Prata, Elder, Ana Fávia, Henrique, Luidi, Ana Lúcia, Talita, Mara, Michele, Whilehm, Jurandir, Paulinho, Gilvan, Xandao, da COPPE/Civil, Eduardo, Carol, CH Costa, Fernanda, Alex, Sidney, da COPPE/Metalurgia, Isabel. Ao meu grande amigo e agora professor da COPPE/Química, Priamo Melo e muitos outros.

Agradeço à CAPES pela bolsa concedida que foi de vital importância para a realização do curso.

Ao 485 (Gal. Osório/Penha - via Fundão).

Sumário

Introdução

O presente trabalho trata do problema de programação linear em dois níveis (PLDN). O objetivo inicial concentrou-se na comparação (em termos de eficiência computacional) de dois algoritmos globais existentes na literatura. O primeiro deles utiliza um método *branch-and-bound* para resolver (PLDN) globalmente. O segundo utiliza uma estratégia de penalidades associada a um algoritmo tipo *outer approximation*, determinando uma ε-solução global de (PLDNP), que é um problema de programação linear em dois níveis onde não existem restrições no primeiro nível.

Foram propostos neste trabalho os algoritmos de penalidade e *branch-and-bound* modificados. Os mesmos contornaram as deficiências apresentadas pelos algoritmos originais nos testes realizados. O trabalho é então divido da seguinte forma:

No capítulo 1 é dada uma introdução ao problema de programação linear em dois níveis. Neste capítulo são apresentadas as principais propriedades e características de (PLDN) que são a base para o desenvolvimento de todos os algoritmos conhecidos, voltados a sua solução global.

A abragência da formulação de (PLDN) também é apresentada, mostrando como problemas usuais de programação matemática podem ser reformulados como um (PLDN). Além disto são citadas algumas aplicações e um problema em particular é apresentado em detalhes, dando uma idéia das vantagens de um modelo linear hierárquico em relação a um modelo linear usual.

No capítulo 2 são apresentados os principais algoritmos existentes na literatura para a resolução de (PLDN) e em particular de (PLDNP). O conhecimento destes algoritmos mostrou-se de fundamental importância, visto que através dos mesmos foi possível uma melhor compreensão dos algoritmos de *branch-and-bound* e penalidade em estudo, e consequentemente do desenvolvimento das modificações introduzidas.

No capítulo 3 são apresentados, em detalhes, os dois algoritmos em estudo. No in-

tuito de facilitar o entendimento, procurou-se descrever os mesmos com uma notação semelhante àquela utilizada no capítulo anterior.

No capítulo 4 são finalmente apresentadas as modificações propostas para os algoritmos de *branch-and-bound* e penalidade originais. Estas modificações basearam-se na substituição do algoritmo *outer approximation*, utilizado no algoritmo de penalidade, por um algoritmo enumerativo que se assemelha em parte a um algoritmo *branch-and-bound*. A segunda modificação introduziu um algoritmo de ponto de equilíbrio, utilizado no algoritmo de penalidade, nos nós da árvore gerada pelo algoritmo *branch-and-bound* original.

Finalmente, no capítulo 5 são apresentados os resultados computacionais obtidos ao se testar os algoritmos originais e modificados, em problemas gerados aleatoriamente. Algumas estatísticas são apresentadas, mostrando a diferença em termos de desempenho computacional, entre os algoritmos originais e modificados. Seguem-se as conclusões e sugestões para trabalhos futuros.

Capítulo 1

Programação Linear em Dois Níveis - PLDN

A programação em dois níveis é uma área da programação matemática que estuda problemas de otimização formulados segundo uma estrutura hierárquica. Esta estrutura é composta por um nível superior (primeiro nível) e um nível inferior (segundo nível). As ações tomadas pelo agente do nível superior restringem as possibilidades de decisão do nível inferior, que por sua vez, também influenciam nas ações do primeiro.

Neste trabalho é estudado um caso particular deste problema de programação matemática, que é o caso linear. Nas próximas seções o problema de programação linear em dois níveis (PLDN) é introduzido. Na seção 1.1 apresentamos a formulação matemática deste problema. Na seção 1.2 a abrangência do problema é apresentada mostrando-se como problemas típicos de programação matemática podem ser formulados como um problema linear em dois níveis. Na seção 1.3 são apresentadas as principais características de PLDN especialmente relacionadas ao conjunto viável do problema. Na seção 1.4 aplicações da programação matemática em dois níveis são citadas, e em particular um problema prático é apresentado em detalhes.

1.1 Definições

O problema de programação matemática em dois níveis, tem como característica principal o fato de que sua região viável é parcialmente definida pelo conjunto de soluções do nível inferior, que por sua vez é parametrizado pelas variáveis de decisão do nível superior. Este problema pode ser formulado da seguinte forma:

Atribui-se ao vetor $x \in \Re^{n_1}$ as n_1 variáveis de decisão do agente do nível superior (líder) e à $y \in \Re^{n_2}$ as n_2 variáveis de decisão do agente do nível inferior (seguidor).

O problema do nível inferior é definido como:

$$P(x) \quad \max\{f_2(x,y) : y \in W_2(x)\}$$

onde $f_2 : \Re^{n_1} \times \Re^{n_2} \to \Re$ é a função objetivo do seguidor e $W_2(x)$ o seu conjunto viável parametrizado pelas variáveis de decisão do líder.

O problema superior é definido em função do conjunto de soluções $Y(x) = \arg\max\{f_2(x,y) : y \in W_2(x)\}$ do problema paramétrico $P(x)$ da seguinte forma:

$$\begin{array}{ll} \max & f_1(x,y) \\ \text{s.a} & (x,y) \in W_1,\ y \in Y(x) \end{array}$$

onde $f_1 : \Re^{n_1} \times \Re^{n_2} \to \Re$ é a função objetivo do líder e $W_1 \subseteq \Re^{n_1} \times \Re^{n_2}$.

Neste trabalho considera-se o caso mais simples da formulação acima, onde as funções f_1 e f_2 são lineares e os conjuntos W_1 e $W_2(x)$ poliedros. Portanto, o problema de programação linear em dois níveis é definido como:

$$\begin{array}{lll} \text{(PLDN)} & \max\limits_{x,y} & f_1(x,y) = c_1^T x + c_2^T y \qquad (1.1) \\ & \text{s.a} & B_1 x + B_2 y \leq b \qquad (1.2) \\ & & x \geq 0,\ y \text{ solução de} \qquad (1.3) \\ & & \max\limits_{y} f_2(x,y) = d^T y \qquad (1.4) \\ & & \text{s.a } A_1 x + A_2 y \leq a \qquad (1.5) \\ & & y \geq 0 \qquad (1.6) \end{array}$$

onde $x, c_1 \in \Re^{n_1}$, $c_2, d, y \in \Re^{n_2}$, $a \in \Re^{m_2}$, $b \in \Re^{m_1}$, $A_1 \in \Re^{m_2 \times n_1}$, $A_2 \in \Re^{m_2 \times n_2}$, $B_1 \in \Re^{m_1 \times n_1}$ e $B_2 \in \Re^{m_1 \times n_2}$. O problema do primeiro nível é definido pelas variáveis de decisão do líder x, sua função objetivo (1.1) e seu conjunto de restrições (1.2)-(1.3). Analogamente o problema do segundo nível é definido pelas variáveis de decisão do seguidor y, sua função objetivo (1.4) e seu conjunto de restrições (1.5)-(1.6).

O problema (PLDN) foi estudado inicialmente na teoria dos jogos, sendo o mesmo semelhante ao problema estático de Stackelberg [49]. Uma revisão bibliográfica detalhada sobre (PLDN) pode ser encontrada em Vicente e Calamai [55].

Neste trabalho considera-se, em particular, o problema (PLDNP) que é o problema (PLDN) omitindo-se as restrições do primeiro nível (1.2). Este problema tem sido bastante estudado na literatura como pode ser visto em Júdice e Faustino [37], Önal [44], White e Anandalingam [56], Amouzegar e Moshirvazari [2], dentre outros.

Associados ao problema (PLDN), os seguintes conjuntos podem ser difinidos:

- Conjunto viável relaxado:

$$W = \{(x, y) \in \Re^{n_1} \times \Re^{n_2} : A_1x + A_2y \leq a, \;\; B_1x + B_2y \leq b, \;\; x \geq 0, \;\; y \geq 0\}.$$

- Conjunto viável do segundo nível para cada $x \in \Re^{n_1}_+$:

$$W(x) = \{y \in \Re^{n_2} : A_2y \leq a - A_1x, \;\; y \geq 0\}.$$

- Conjunto solução do segundo nível para cada $x \in \Re^{n_1}_+$:

$$Y(x) = \arg\max\{d^Ty : y \in W(x)\}.$$

- Conjunto viável de PLDN:

$$V = \{(x, y) \in W : y \in Y(x)\}.$$

- Conjunto solução de PLDN:

$$V^\star = \arg\max\{c_1^Tx + c_2^Ty : (x, y) \in V\}.$$

Visto que $V \subseteq W$, o problema relaxado (RPLDN) pode ser definido como: $\max\{c_1^Tx + c_2^Ty : (x, y) \in W\}$ e $W^\star = \arg\max\{c_1^Tx + c_2^Ty : (x, y) \in W\}$ o seu conjunto solução.

Analogamente, o problema relaxado de (PLDNP) chamado (RPLDNP) e seu conjunto solução são definidos da mesma forma, onde o conjunto W é dado simplesmente por:

$$W = \{(x, y) \in \Re^{n_1} \times \Re^{n_2} : A_1x + A_2y \leq a, \;\; x \geq 0, \;\; y \geq 0\}.$$

1.2 Abrangência de PLDN

Esta seção tem como objetivo mostrar como importantes problemas de programação matemática podem ser modelados como problemas de programação linear em dois níveis.

1.2.1 Programação linear 0-1 mista

Seguindo-se a formulação descrita por Audet *et al.* [4], o problema de programação linear 0-1 mista é modelado como:

$$\begin{array}{lll} \text{(PLIM)} & \max & c_1^T x + c_2^T y \\ & \text{s.a} & B_1 x + B_2 y \leq b \\ & & x \geq 0,\ y \in \{0,1\}^p \end{array}$$

onde $\{0,1\}^p$ representa o conjunto dos vetores p-dimensionais com componentes 0 ou 1, c_1, c_2, b, x, y são vetores e B_1, B_2 matrizes.

A seguinte reformulação de (PLIM) em um problema linear em dois níveis sugere que um (PLDN) seja no mínimo tão difícil de ser resolvido quanto um problema linear inteiro (Audet *et al.*[4, seção 5]).

Introduzindo-se no modelo linear anterior uma variável auxiliar $\nu \in \Re^p$, as variáveis binárias $y_i \in \{0,1\}, \forall i \in \{1,2,..,p\}$ podem ser convertidas em variáveis contínuas do tipo $0 \leq y_i \leq 1, \forall i \in \{1,2,..,p\}$ e o modelo inicial convertido no PLDN abaixo:

$$\begin{array}{lll} \text{(PLDN1)} & \max\limits_{x,y,\nu} & c_1^T x + c_2^T y \\ & \text{s.a} & B_1 x + B_2 y \leq b \\ & & x \geq 0, 0 \leq y \leq e \\ & & \nu = 0,\ \nu \text{ solução de} \\ & & \max\limits_{\nu}\ e^T \nu \\ & & \text{s.a}\ \ \nu \leq y \\ & & \qquad \nu \leq e - y \\ & & \qquad \nu \geq 0 \end{array}$$

onde $e^T = (1, 1, \ldots, 1) \in \Re^p$.

Observe que a restrição $\nu = 0$ imposta no primeiro nível de (PLDN1) obriga que um dado ponto $(x,y) \in \{B_1 x + B_2 y \leq b, x \geq 0, 0 \leq y \leq e\}$ seja viável para (PLDN1) apenas quando $y \in \{0,1\}^p$, condição esta garantida pelo conjunto solução do seguidor $\nu = \arg\max\{e^T \nu : \nu \leq y, \nu \leq e - y, \nu \geq 0\} = 0$.

Este fato implica que um ponto (x,y) será viável para (PLIM) se, e somente se, $(x,y,0)$ for viável para (PLDN1). Adicionalmente, como os dois problemas possuem a mesma função objetivo, conclui-se que ambos terão soluções ótimas correspondentes.

1.2.2 Complementaridade linear generalizada

Considere o seguinte problema general de complementaridade linear considerado por Júdice e Mitra [38]:

$$\begin{aligned} (\text{PLRC}) \quad \max \quad & c_1^T x_1 + c_2^T x_2 + c_3^T x_3 && (1.7)\\ \text{s.a} \quad & Q_1 x_1 + Q_2 x_2 + Q_3 x_3 = q && (1.8)\\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 && (1.9)\\ & x_2^T x_3 = 0 && (1.10) \end{aligned}$$

onde $c_1, c_2, c_3, q, x_1, x_2, x_3$ são vetores e Q_1, Q_2, Q_3 matrizes.

Analogamente ao problema linear inteiro anterior, onde as restrições de integralidade foram substituídas por um problema linear, as restrições de complementaridade são também substituídas, analogamente, por um problema linear com a introdução da variável $\nu \in \Re^p$. O problema é então reformulado no seguinte problema linear em dois níveis:

$$\begin{aligned} (\text{PLDN2}) \quad \max_{x,\nu} \quad & c_1^T x_1 + c_2^T x_2 + c_3^T x_3\\ \text{s.a} \quad & Q_1 x_1 + Q_2 x_2 + Q_3 x_3 = q\\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0\\ & \nu = 0,\ \nu \text{ solução de}\\ & \max_{\nu} e^T \nu\\ & \text{s.a } \nu \leq x_2\\ & \qquad \nu \leq x_3\\ & \qquad \nu \geq 0 \end{aligned}$$

onde $e^T = (1, 1, \ldots, 1) \in \Re^p$.

Observe que, analogamente ao problema linear inteiro anterior, a restrição $\nu = 0$ imposta no primeiro nível de (PLDN2) obriga que um dado ponto $(x_1, x_2, x_3) \in \{Q_1 x_1 + Q_2 x_2 + Q_3 x_3 = q, x_1 \geq 0, x_2 \geq 0, x_3 \geq 0\}$ seja viável para (PLDN2) apenas quando $x_2^T x_3 = 0$, condição esta garantida pelo conjunto solução do seguidor $\nu = \arg\max\{e^T \nu : \nu \leq x_2, \nu \leq x_3, \nu \geq 0\} = 0$.

Este fato implica que um ponto (x_1, x_2, x_3) será viável para (PLRC) se, e somente se, $(x_1, x_2, x_3, 0)$ for viável para (PLDN2). A correspondência entre os conjuntos viáveis de ambos os problemas e o fato das funções objetivo serem as mesmas leva a conclusão de que ambos os problemas terão solucões ótimas correspondentes.

1.2.3 Programação linear max-min

O problema de programação max-min linear é um caso particular de PLDNP. Sua formulação geral, dada por Falk [25], é apresentada abaixo:

$$\text{(PMM)} \quad \max_{x\geq 0} \min_{y\geq 0}\{c_1^T x + c_2^T y : A_1 x + A_2 y \leq a\}$$

onde c_1, c_2, a, x, y são vetores e A_1, A_2 matrizes. Observe que (PMM) pode ser alternativamente escrito de duas maneiras equivalentes:

$$\text{(PMM)} \quad \max_{x\geq 0} \left\{c_1^T x + \min_{y\geq 0}\{c_2^T y : A_2 y \leq a - A_1 x\}\right\}$$

$$\text{(PMM)} \quad \max_{x\geq 0} \left\{c_1^T x + c_2^T y : y \in \arg\min\{c_2^T y : A_2 y \leq a - A_1 x, y \geq 0\}\right\}$$

Finalmente, esta última formulação alternativa permite a clara visualização de (PMM) como o seguinte (PLDNP):

$$\begin{array}{lll} \text{(PLDNP1)} & \max\limits_{x,y} & c_1^T x + c_2^T y \\ & \text{s.a} & x \geq 0,\ y \text{ solução de} \\ & & \max\limits_{y} \ -c_2^T y \\ & & \text{s.a} \ \ A_2 y \leq a - A_1 x \\ & & \qquad y \geq 0 \end{array}$$

1.2.4 Programação bilinear

O problema de programação bilinear (PBL) tem sido bastante estudado na literatura, como pode ser visto nos trabalhos de Konno [40], Vaish e Shetty [54], Sherali e Shetty [47], Alarie *et al.* [1], Audet *et al.* [5], dentre outros. Segundo Audet *et al.* [5] o problema (PBL) é dado pela seguinte formulação:

$$\begin{array}{lll} \text{(PBL)} & \max & c^T x - u^T Q x + u^T d \\ & \text{s.a} & x \in X, u \in U \end{array}$$

onde $X = \{x \in \Re^{n_x} : Ax \leq a, x \geq 0\}$ e $U = \{u \in \Re^{n_u} : u^T B \leq b^T, u \geq 0\}$. Os vetores a, b, c, d e as matrizes A, B, Q possuem dimensões apropriadas.

Muitos autores têm observado que o problema (PBL) pode ser reformulado como o seguinte problema de otimização: $\max\{f(x) : x \in X\}$, onde a função $f : \Re^{n_x} \to \Re$, definida abaixo, é linear por partes e convexa visto que é definida pelo máximo de uma família de funções lineares (Rockafellar [46, teorema 5.5]).

$$f(x) = c^T x + \max\{u^T(d - Qx) : u \in U\}$$

Assumindo-se que o conjunto U seja não vazio, pela teoria da dualidade o problema acima pode ser reformulado como:

$$f(x) = c^T x + \min\{b^T y : By \geq d - Qx, y \geq 0\}$$

Fazendo-se uso desta formulação, o problema (PBL) pode então ser reformulado como o seguinte problema de programação max-min linear (seção 1.2.3):

$$\max_{x \in X}\{c^T x + b^T y : y \in \arg\min\{b^T y : By \geq d - Qx, y \geq 0\}\}$$

Esta nova formulação permite a clara visualização de (PBL) como o seguinte (PLDN):

$$\begin{array}{lll} \text{(PLDN3)} & \max\limits_{x,y} & c^T x + b^T y \\ & \text{s.a} & Ax \leq a,\ x \geq 0,\ y \text{ solução de} \\ & & \max\limits_{y} -b^T y \\ & & \text{s.a } By \geq d - Qx \\ & & \quad y \geq 0 \end{array}$$

Quando o conjunto U é vazio, o problema do segundo nível em PLDN3 pode ser inviável ou ilimitado, pois ele corresponde ao dual de um problema linear inviável. De qualquer modo, PLDN3 seria inviável, o que manteria a equivalência com PBL.

1.3 Características de PLDN

Esta seção tem por objetivo enumerar as principais características de PLDN, referentes a vários aspectos como convexidade, conexidade, relações com o conjunto viável relaxado, características de suas soluções e sua complexidade.

1.3.1 Conjunto viável

A primeira propriedade de PLDN, citada por vários autores e apresentada em exemplos como em Bialas e Karwan [13], diz respeito a não convexidade de seu conjunto viável.

Proposição 1.3.1 *O conjunto viável V de PLDN é, em geral, um conjunto não convexo.*

Sendo o conjunto V não convexo, uma característica importante que pode ser deduzida é a de que PLDN pode possuir ótimos locais.

O conjunto V, apesar de não convexo, é a união de um número finito de conjuntos convexos como mostrado abaixo por Campêlo [16, teorema 1.2.1], que generalizou para PLDN os resultados obtidos anteriormente por Benson [12, teorema 3.2], para PLDNP e PLDN quando $B_2 = \emptyset$.

Propriedade 1.3.1 *Sejam W_i, $i \in I$, as faces de W. Então existe $I' \subseteq I$ tal que $V = \bigcup_{i \in I'} W_i$, ou seja, o conjunto viável de PLDN é união de faces do conjunto viável relaxado.*

Sendo o conjunto convexo W um poliedro, as suas faces também são poliedrais (Rockafellar [46, pág. 162]), sendo portanto conjuntos fechados. Deste resultado conclui-se que o conjunto V também é um conjunto fechado, pois é uma união de um número finito de conjuntos fechados.

Definidas as relações entre o conjunto V e o conjunto W, uma propriedade importante obtida por Campêlo [16, teorema 1.2.2] mostra que o conjunto viável V, embora não convexo, pode ser substituído por sua envoltória convexa (co) na definição de PLDN.

Propriedade 1.3.2 *Associado a PLDN, seja definido o seguinte problema convexo*

$$PLDN_c \quad \max \; f_1(x,y) \quad \text{s.a} \quad (x,y) \in \text{co}\, V.$$

Os problemas PLDN e $PLDN_c$ são simultaneamente inviáveis, ilimitados ou têm solução. Neste último caso, tem-se ainda que:

(i) Toda solução de PLDN é solução de $PLDN_c$.

(ii) Toda solução extrema de PLDN é solução extrema de $PLDN_c$ e vice-versa.

Sendo o conjunto viável V em geral não convexo, pode-se então indagar sobre a sua **conexidade**. A definição de conexidade e a propriedade seguinte são encontradas em Benson [12].

Definição 1.3.1 *Um conjunto C é dito* **conexo** *quando não pode ser escrito como a união de dois conjuntos A e B não vazios e abertos e tais que*

$$A \cap (cl\, B) = \emptyset \quad e \quad (cl\, A) \cap B = \emptyset,$$

onde cl denota o fecho do conjunto.

Segundo Campêlo [16] o conceito de conexidade de conjuntos está ligado à idéia de continuidade. De acordo com o mesmo, um conjunto C é conexo quando existe sempre um "caminho contínuo" de pontos de C entre qualquer par de pontos deste conjunto.

Propriedade 1.3.3 *Se as restrições do primeiro nível independem da variável do segundo nível, isto é, $B_2 = 0$, então o conjunto viável V de PLDN é conexo e em geral não convexo.*

1.3.2 Soluções de PLDN

Esta breve seção apresenta uma das propriedades mais importantes de PLDN, que caracteriza as suas soluções. Esta propriedade tem sido de fundamental importância para o desenvolvimento de algoritmos que procuram resolver PLDN através da pesquisa de vértices do conjunto viável relaxado W.

Além disto ela possibilita a adequação de métodos usuais de programação linear como ferramentas para a solução de PLDN. Este resultado pode ser obtido de Candler e Townsley [20] e Bialas e Karwan [13] para PLDNP e de Campêlo [16, colorário 1.2.2] para PLDN.

Propriedade 1.3.4 *Se PLDN tem solução, pelo menos uma delas é atingida em um vértice do conjunto viável relaxado W.*

Ressalta-se também que equivalemente aos ótimos globais, os ótimos locais de PLDN também são atingidos em vértices do conjunto viável relaxado W.

1.3.3 Complexidade

O objetivo desta seção é dar uma idéia da dificuldade de resolução de PLDN. Na seção 1.2.1 foi mostrado que PLDN pode ser reformulado como um problema de programação linear 0-1 mista. Baseando-se nesta constatação é fácil perceber que a resolução de PLDN seja no mínimo tão difícil quanto a resolução de um problema linear inteiro.

Um dos primeiros autores a estudar a complexidade de PLDN foi Jeroslow [36] que mostrou que PLDN é um problema NP-Difícil. Um pouco mais tarde Ben-Ayed e Blair [9] mostraram que o problema de programação multinível, cujo PLDN é um caso particular, é um problema NP-Difícil através do conhecido problema da mochila (*Knapsack Problem*).

Em [29], Hansen *et al.* mostram que nenhum esquema de aproximação polinomial para se obter uma ε-solução para PLDN em tempo polinomial, no tamanho do problema e no valor de ε, pode ser obtido. Os autores deduziram este resultado ao mostrar que o problema de programação max-min linear (seção 1.2.3), caso particular de PLDN, é fortemente NP-Difícil.

1.4 Aplicações

Esta seção tem por objetivo apresentar algumas aplicações práticas da programação matemática em dois níveis. As áreas de abrangência são das mais variadas, desde a engenharia até a economia. Dentre muitas aplicações conhecidas na literatura, destacam-se as seguintes.

Na engenharia podem ser citadas aplicações no projeto de estruturas metálicas (Li *et al.* [41]), em estruturas de concreto protendido (Kirsch [39]), no dimensionamento ótimo de sólidos elásticos em contato (Herskovits *et al.* [31]).

Na área de transportes citam-se trabalhos em transporte urbano (Clegg *et al.* [23]), em tráfego (Yang e Yagar [58] e Migdalas [42]), na estimação de matrizes de demanda de fluxo entre pontos origem e destino (Florian e Chen [22, 26, 27]).

Na área de projeto de redes (*network design problem*) citam-se os trabalhos de Ben-Ayed *et al.* [10, 11]. Ainda na área de redes, Zhang e Zhu [59] e Wolf e Smeers [57] usam programação em dois níveis no dimensionamento ótimo de redes de tubulações.

A programação em dois níveis também é empregada na determinação de políticas ótimas como mostram trabalhos no setor agrícola (Candler *et al.* [19] e Önal *et al.* [45]) e no setor energético (Hobbs e Nelson [32], Islam [35] e Bard *et al.* [7]).

O trabalho proposto por Bard *et al.* [7] utiliza um modelo em dois níveis para determinar taxas de crédito ótimas para o incentivo à produção de bio-combustível. O modelo em dois níveis por eles apresentado é uma extensão de PLDN no qual

as funções objetivo do líder e do seguidor são bilineares. Maiores detalhes sobre métodos de resolução para este problema podem ser obtidos em [24].

Com o objetivo de fornecer ao leitor uma visão mais abrangente da programação em dois níveis em termos de sua aplicabilidade, na próxima subseção é apresentado com maiores detalhes um problema prático proposto por Amouzegar e Moshirvazari [3] para a determinação de políticas ótimas para o controle de poluição causada por resíduos tóxicos provenientes de indústrias.

1.4.1 Determinação de políticas ótimas para o controle de poluição

Nessa subseção apresentamos um problema de controle de poluição, proveniente de resíduos tóxicos produzidos por indústrias de vários tipos, nos Estados Unidos da América (EUA). O trabalho foi realizado por Amouzegar e Moshirvazari [3] e baseou-se em dados reais obtidos do governo da Califórnia.

Nesse artigo foram propostos dois modelos de otimização. No primeiro deles, um conjunto de indústrias, pertencentes a um conjunto de regiões, tenta minimizar seus custos referentes aos gastos necessários ao tratamento de seus resíduos, sem ter nenhuma intervenção governamental. No segundo modelo, o governo interage com estas indústrias através de impostos cobrados pela produção e tratamento destes resíduos.

O modelo linear convencional

As características referentes ao primeiro modelo são dadas abaixo:

Seja um conjunto de regiões I. Cada região possui um conjunto de n_f diferentes tipos de indústrias. Cada indústria, por sua vez, produz p_w diferentes tipos de resíduos tóxicos, de agora em diante chamado de *RTox*. Estes resíduos podem ser tratados da seguinte forma: redução de sua produção na própria indústria, reciclagem e incineração. A reciclagem pode ser usada apenas em um subconjunto $R(w)$ de *RTox*. Além disso, sua eficiência não é 100% e o produto restante deve ser incinerado.

Cada indústria tem a opcão de construir seus próprios recicladores ou então juntamente com outras indústrias construir grandes recicladores de uso comum. É

importante salientar que um reciclador pode ser usado para tratar mais de um tipo de resíduo tóxico. Uma outra opção é a construção de grandes incineradores de uso comum.

O modelo linear proposto (Figura 1.1) considera apenas as indústrias agindo em grupo com o objetivo de minimizar seus custos. A função objetivo minimiza os custos de transporte necessário para se deslocar determinada quantidade de resíduos de uma região i para uma região j para ser reciclado ou incinerado, os custos de incineração, os custos de reciclagem, os custos de instalação de recicladores na própria indústria e os custos de instalação de recicladores e incineradores de uso comum em determinada região i.

Neste modelo as indústrias existentes no conjunto de regiões I além de decidirem sobre a quantidade de seus resíduos que devem ser enviados para os recicladores ou incineradores, também decidem sobre a alocação de incineradores e recicladores de uso comum em uma dada região i. A seguir são definidos os índices, conjuntos, parâmetros e variáveis de decisão utilizadas no modelo linear proposto.

	Índices e Conjuntos
i	Índice das regiões contidas no conjunto de regiões I, $(i \in I)$;
f	Índice dos tipos de indústrias contidas no conjunto de tipos de indústrias $F = \{1, \ldots, n_f\}$, $(f \in F)$;
r	Índice dos tipos de recicladores contidos no conjunto de tipos de recicladores $R = \{1, \ldots, q_r\}$, $(r \in R)$;
d	Índice dos tipos de incineradores contidos no conjunto de tipos de incineradores $D = \{1, \ldots, m_d\}$, $(d \in D)$;
w	Índice dos tipos de *RTox* contidos no conjunto de tipos de *RTox* $W = \{1, \ldots, p_w\}$, $(w \in W)$;
$R(w)$	Subconjunto dos tipos de *RTox* w que podem ser reciclados;
	Parâmetros
A_{wi}	Quantidade (em ton.) de *RTox* do tipo w produzido na região i;
α_{wfi}	Fração de *RTox* tipo w produzido por uma indústria tipo f localizada na região i;
β_{wr}	Eficiência do reciclador tipo r na reciclagem de um *RTox* tipo w;
$Rcap_r$	Capacidade (ton.) do reciclador tipo r localizado em uma indústria f;
$Ocap_r$	Capacidade (ton.) do reciclador de uso comun tipo r;
$Icap_d$	Capacidade (ton.) do incinerador tipo d;
M_{ij}	Distância (em Km) entre a região i e a região j;
IC_{wd}	Custo para incinerar 1 ton. de *RTox* tipo w em um incinerador tipo d;
RC_{wr}	Custo para reciclar 1 ton. de resíduo tipo w em um reciclador tipo r;
FR_r	Custo de instalação de um reciclador do tipo r;
FI_d	Custo de instalação de um incinerador do tipo d;
TC_{ij}	Custo de transporte (\$/Km) de *RTox* da região i para a região j;
	Variáveis de Decisão
xn_{wifr}	Quantidade de resíduo tóxico do tipo w reciclado em um reciclador do tipo r pertencente a uma indústria do tipo f localizada em uma região i;
xo_{wifjr}	Quantidade de resíduo tóxico do tipo w transportado de uma região i para uma região j por uma indústria do tipo f para ser reciclado em um reciclador de uso comun do tipo r;
y_{wifjd}	Quantidade de resíduo tóxico do tipo w transportado de uma região i para uma região j por uma indústria do tipo f para ser incinerado em um incinerador tipo d;
yn_{wifjd}	Quantidade de lixo proveniente da reciclagem de resíduo tóxico do tipo w transportado de uma região i para uma região j por uma indústria do tipo f para ser incinerado em um incinerador de uso comum do tipo d;
yo_{wijd}	Quantidade de lixo proveniente da reciclagem de resíduo tóxico do tipo w reclicado em um reciclador de uso comun, transportado de uma região i para uma região j para ser incinerado em um incinerador tipo d;
p_{ifr}	Número de recicladores do tipo r construídos por uma indústria do tipo f em uma região i;
o_{ir}	Número de recicladores tipo r de uso comun instalados em uma região i;
q_{id}	1, se uma região i possui um incinerador do tipo d, 0 caso contrário;

$$min \quad \sum_{i\in I}\sum_{f\in F}\left\{\sum_{k\in I}\sum_{d\in D}\left[\sum_{w\in W}(TC_{ik}+IC_{wd})\,y_{wifkd}+\sum_{w\in R(w)}(TC_{ik}+IC_{wd})\,yn_{wifkd}\right]+\sum_{w\in R(w)}\sum_{r\in R}\sum_{j\in I}(TC_{ij}+RC_{wr})\,xo_{wifjr}\right\}+$$

$$\sum_{i\in I}\sum_{f\in F}\sum_{r\in R}FR_r\cdot p_{ifr}+\sum_{i\in I}\sum_{r\in R}FR_r\cdot o_{ir}+\sum_{i\in I}\sum_{d\in D}FI_d\cdot q_{id}+\sum_{w\in R(w)}\sum_{i\in I}\sum_{f\in F}\sum_{r\in R}RC_{wr}\cdot xn_{wifr}+$$

$$\sum_{w\in R(w)}\sum_{j\in I}\sum_{k\in I}\sum_{d\in D}(TC_{jk}+IC_{wd})\,yo_{wjkd}$$

$$s.a: \quad \sum_{r\in R}\left(\sum_{w\in R(w)}xn_{wifr}-p_{ifr}\cdot Rcap_r\right)\leq 0,\forall i\in I,f\in F \tag{1}$$

$$\sum_{w\in R(w)}\sum_{i\in I}\sum_{f\in F}xo_{wifjr}\leq o_{jr}\cdot Ocap_r,\forall j\in I,r\in R \tag{2}$$

$$\sum_{i\in I}\left[\sum_{w\in R(w)}\left(yo_{wijd}+\sum_{f\in F}yn_{wifjd}\right)+\sum_{w\in W}\sum_{f\in F}y_{wifjd}\right]\leq q_{jd}\cdot Icap_d,\forall j\in I,d\in D \tag{3}$$

$$\sum_{k\in I}\sum_{d\in D}yo_{wfkd}-\sum_{i\in I}\sum_{f\in F}\sum_{r\in R}\beta_{wr}\cdot xo_{wifjr}=0,\forall w\in R(w),j\in I \tag{4}$$

$$\sum_{k\in I}\sum_{d\in D}yn_{wifkd}-\sum_{r\in R}\beta_{wr}\cdot xn_{wifr}=0,\forall w\in R(w),i\in I,f\in F \tag{5}$$

$$\sum_{r\in R}\left(xn_{wifr}+\sum_{j\in I}xo_{wifjr}\right)+\sum_{k\in I}\sum_{d\in D}y_{wifkd}=\alpha_{wif}\cdot A_{wi},\forall w\in R(w),i\in I,f\in F \tag{6}$$

$xn\geq 0$, $xo\geq 0$, $y\geq 0$, $yn\geq 0$, $yo\geq 0$, p inteiro positivo, o inteiro positivo, $q\in\{0,1\}\in\Re^{|I|}$ (7)

Figura 1.1: Modelo linear convencional

O modelo linear em dois níveis

Os resultado obtidos com o modelo linear anterior, tomando-se como parâmetros dados reais fornecidos pelo governo da Califórnia, indicaram como cenário ótimo a instalação de grandes incineradores em determinadas regiões. Este resultado era esperado visto que o preço de incineração é extremamente baixo e supõe-se que o incinerador pode trabalhar em capacidade máxima.

Este cenário, apesar de ser ótimo do ponto de vista do conjunto de indústrias, é extremamente negativo do ponto de vista social. Grandes incineradores podem causar sérios problemas ambientais. A necessidade da interveção do governo adotando políticas restritivas as ações destas indústrias se torna claro.

O primeiro modelo é então convertido em um problema de dois níveis (Figura 1.2) onde o governo encontra-se no nível superior sendo o líder e o conjunto de indústrias sendo o seguidor. Neste modelo o governo intervem com taxas cobradas pela utilização de incineradores e recicladores de uso comum para tratarem determinado tipo de resíduo tóxico, além disto são cobradas taxas sobre a produção de cada tipo de resíduo. O governo também tem o controle sobre o número de incineradores e recicladores que podem ser instalados em cada região i. O seguidor (indústrias) tem o controle de alocação, apenas sobre seus próprios recicladores e suas respectivas capacidades. As variáveis de decisão do problema são as mesmas do problema anterior, acrescentando-se apenas algumas variáveis referentes as taxas cobradas pelo governo. As retrições do modelo também permaneceram inalteradas.

Variáveis atribuídas ao líder: μ_{wd}, ν_{wr}, τ_w, yo_{wijd}, o_{ir}, q_{id}	
onde:	
μ_{wd}	Preço unitário cobrado para incinerar um resíduo tóxico do tipo w em um incinerador do tipo d;
ν_{wr}	Preço unitário cobrado para reciclar um resíduo tóxico do tipo w em um reciclador de uso comum do tipo r;
τ_w	Imposto cobrado pela produção de resíduo sólido do tipo w;
Variáveis atribuídas ao seguidor: xn_{wifr}, xo_{wifjr}, y_{wifjd}, yn_{wifjd}, p_{ifr}	

O resultado ótimo mostrou-se extremamente satisfatório do ponto de vista social. O cenário obtido indicou um aumento no número de recicladores de uso comum e no número de recicladores pertencentes a cada indústria, além disto as taxas cobradas geraram um lucro de US$ 470.000 ao ano para o governo.

$$\min \quad \sum_{i\in I}\sum_{f\in F}\left\{\sum_{k\in I}\sum_{d\in D}\left[\sum_{w\in W}(TC_{ik}+IC_{wd})\,y_{wifkd}+\sum_{w\in R(w)}(TC_{ik}+IC_{wd})\,yn_{wifkd}\right]+\sum_{w\in R(w)}\sum_{r\in R}\sum_{j\in I}(TC_{ij}+RC_{wr})\,xo_{wifjr}\right\}+$$
$$\sum_{i\in I}\sum_{f\in F}\sum_{r\in R}FR_r\cdot p_{ifr}+\sum_{i\in I}\sum_{r\in R}FR_r\cdot o_{ir}+\sum_{i\in I}\sum_{d\in D}FI_d\cdot q_{id}+\sum_{w\in R(w)}\sum_{i\in I}\sum_{f\in F}\sum_{r\in R}RC_{wr}\cdot xn_{wifr}+$$
$$\sum_{w\in R(w)}\sum_{j\in I}\sum_{k\in I}\sum_{d\in D}(TC_{jk}+IC_{wd})\,yo_{wjkd}$$

$\mu\geq 0,\nu\geq 0,\tau\geq 0,yo\geq 0$, o inteiro positivo, $q\in\{0,1\}\in\Re^{|I|}$, e $[xn^T xo^T y^T yn^T p^T]$ solução de:

$$\min \quad \sum_{w\in R(w)}\sum_{i\in I}\sum_{f\in F}\left[\sum_{r\in R}RC_{wr}\cdot xn_{wifr}+\sum_{j\in I}\sum_{d\in D}(\mu_w+\tau_w+TC_{ij})\,y_{wifjd}\right]+\sum_{i\in I}\sum_{f\in F}\sum_{r\in R}FR_r\cdot p_{ij}+$$
$$\sum_{w\in R(w)}\sum_{i\in I}\sum_{f\in F}\left[\sum_{j\in I}\left(\sum_{d\in D}(\mu_w+TC_{if})\,yn_{wifjd}+(\nu_w+TC_{ij})\sum_{r\in R}xo_{wifjr}\right)\right]$$

$$s.a: \quad \sum_{r\in R}\left(\sum_{w\in R(w)}xn_{wifr}-p_{ifr}\cdot Rcap_r\right)\leq 0,\forall i\in I,f\in F \tag{1}$$
$$\sum_{w\in R(w)}\sum_{i\in I}\sum_{f\in F}xo_{wifjr}\leq o_{jr}\cdot Ocap_r,\forall j\in I,r\in R \tag{2}$$
$$\sum_{i\in I}\left[\sum_{w\in R(w)}\left(yo_{wijd}+\sum_{f\in F}yn_{wifjd}\right)+\sum_{w\in W}\sum_{f\in F}y_{wifjd}\right]\leq q_{jd}\cdot Icap_d,\forall j\in I,d\in D \tag{3}$$
$$\sum_{k\in I}\sum_{d\in D}yo_{wfkd}-\sum_{i\in I}\sum_{f\in F}\sum_{r\in R}\beta_{wr}\cdot xo_{wifjr}=0,\forall w\in R(w),j\in I \tag{4}$$
$$\sum_{k\in I}\sum_{d\in D}yn_{wifkd}-\sum_{r\in R}\beta_{wr}\cdot xn_{wifr}=0,\forall w\in R(w),i\in I,f\in F \tag{5}$$
$$\sum_{r\in R}\left(xn_{wifr}+\sum_{j\in I}xo_{wifjr}\right)+\sum_{k\in I}\sum_{d\in D}y_{wifkd}=\alpha_{wif}\cdot A_{wi},\forall w\in R(w),i\in I,f\in F \tag{6}$$
$$xn\geq 0,xo\geq 0,y\geq 0,yn\geq 0,p \text{ inteiro positivo} \tag{7}$$

Figura 1.2: Modelo linear em dois níveis

Capítulo 2

Métodos de resolução

Este capítulo se dedica a apresentação dos principais algoritmos existentes na literatura para a resolução do PLDN. Para cada algoritmo apresentado é dada uma visão geral sobre as suas principais características como, por exemplo, as hipótesis consideradas, formulações equivalentes utilizadas e estratégias de resolução.

A quase totalidade dos algoritmos existentes utilizam ferramentas de programação linear para resolver PLDN através da busca de pontos extremos do conjunto viável relaxado W. Na seção 1.3.3 a complexidade de PLDN foi apresentada, mostrando-se ser um problema fortemente NP-Difícil e que, devido a este fato, não existe algoritmo que possa resolver globalmente PLDN em tempo polinomial, a menos que P≡NP.

Os principais algoritmos existentes na literatura, que se baseiam na busca de pontos extremos do conjunto viável relaxado W, podem ser agrupados em: algoritmos de penalidade (2.1), algoritmos tipo *branch-and-bound* (2.2), complementaridade linear (2.3) e enumeração de pontos extremos (2.4).

Ressalta-se, entretanto, que além destes algoritmos ainda existem outros que se baseiam em métodos de pontos interiores, dando-se como exemplo o algoritmo de Herskovits e Leontiev [30].

2.1 Métodos de penalidade

Os algoritmos apresentados nesta seção são voltados à resolução global de PLDNP trabalhando com formulações equivalentes.

A seguir são apresentadas duas formulações (P_1) e (P_2) equivalentes ao PLDNP. Em (P_1) o problema do seguidor é substituído por suas condições de Karush-Kuhn-

Tucker (KKT) e em (P_2) é apresentada uma forma alternativa de expressar (P_1). Estas formulações, apesar de eliminarem o problema do seguidor, tornam não convexa a região viável deste novo problema.

Nos modelos (P_1) e (P_2), denota-se por $w \in \Re^{m_2}$ a variável de folga associada às restrições do segundo nível, e por $u \in \Re^{m_2}$ e $v \in \Re^{n_2}$, respectivamente, as variáveis dual e de folga associadas ao problema dual do seguidor.

Os problemas (P_1) e (P_2) são então formulados como:

$$
\begin{array}{lll}
(P_1) & \max & c_1^T x + c_2^T y \\
& \text{s.a} & A_1 x + A_2 y + w = a \\
& & x \geq 0, y \geq 0, w \geq 0 \\
& & A_2^T u - v = d \\
& & u \geq 0, v \geq 0 \\
& & v^T y = u^T w = 0
\end{array}
\qquad
\begin{array}{lll}
(P_2) & \max & c_1^T x + c_2^T y \\
& \text{s.a} & A_1 x + A_2 y \leq a \\
& & A_2^T u \geq d \\
& & u^T(a - A_1 x) - d^T y = 0 \\
& & x \geq 0, y \geq 0, u \geq 0
\end{array}
$$

A expressão $u^T(a - A_1 x) - d^T y$ representa o *gap* de dualidade entre os problemas primal e dual do seguidor, que deve ser nulo em uma solução viável para PLDNP. As restrições $u^T(a - A_1 x) - d^T y = 0$ e $v^T y = u^T w = 0$, são equivalentes, pois esta pode ser reformulada como $u^T w + v^T y = 0$, de maneira que ao se substituir as variáveis de folga u e v obtém-se aquela.

A equivalência entre (P_1), (P_2) e PLDNP se dá no sentido de que um ponto (x, y) é solução global de PLDNP se, e somente se, (x, y, w, u, v) e (x, y, u) são soluções de (P_1) e (P_2) respectivamente, para algum u, w e v (Audet *et al.* [4, proposição 3.3]).

2.1.1 Penalização de folgas complementares

O seguinte algoritmo, proposto por Önal [44], propõe-se a resolver PLDNP globalmente trabalhando com sua formulação equivalente (P_1).

Movendo-se em (P_1) as restrições de complementaridade para a função objetivo do líder, penalizadas através de um parâmetro $M \geq 0$, o seguinte problema é obtido:

$$
\begin{array}{lll}
P(M) & \max & F_M(x, y, w, u, v) = c_1^T x + c_2^T y - M(v^T y + u^T w) \qquad (2.1) \\
& \text{s.a} & A_1 x + A_2 y + w = a \qquad (2.2) \\
& & x \geq 0, y \geq 0, w \geq 0 \qquad (2.3) \\
& & A_2^T u - v = d \qquad (2.4) \\
& & u \geq 0, v \geq 0 \qquad (2.5)
\end{array}
$$

O algoritmo de Önal propõe-se a achar ε-soluções globais de PLDNP através da busca de soluções locais de $P(M)$. O autor introduz a seguinte definição de *solução local estável* de $P(M)$ e mostra que o termo de penalidade se anula em um solução deste tipo (Önal [44, proposição 1]).

Definição 2.1.1 *Uma* **solução local estável** *do problema penalizado $P(M)$ é uma solução viável tal que (i) maximiza localmente (2.1), e (ii) tanto o ponto solução quanto o valor da função objetivo não se alteram para qualquer $M > M_0$, para algum M_0 suficientemente grande.*

O algoritmo de Önal determina uma solução local estável de $P(M)$ utilizando uma adaptação do algoritmo de Beale [8], que encontra ótimos locais de problemas quadráticos. Este método é semelhante ao método simplex com uma pequena adaptação para tratar o termo quadrático em 2.1. Esta adaptação ocorre na escolha da variável a entrar na base, que é determinada em função das derivadas parciais $(\frac{\partial F_M}{\partial x_N}, \frac{\partial F_M}{\partial y_N}, \frac{\partial F_M}{\partial w_N}, \frac{\partial F_M}{\partial u_N}, \frac{\partial F_M}{\partial v_N})$, de F_M em um dado ponto $(\bar{x}, \bar{y}, \bar{w}, \bar{u}, \bar{v})$, com relação às variáveis não-básicas. O parâmetro M é considerado implicitamente como um valor dominante.

Após econtrado um ótimo local $(\hat{x}, \hat{y}, \hat{w}, \hat{u}, \hat{v})$ de $P(M)$, uma restrição do tipo: $c_1^T x + c_2^T y \geq c_1^T \hat{x} + c_2^T \hat{y} + \varepsilon$, onde $\varepsilon \geq 0$ é uma tolerância aceitável, é adicionada ao conjunto de restrições de $P(M)$, gerando o problema $P'(M)$. O método adaptado de Beale é então aplicado a este novo problema.

Caso a solução obtida seja uma solução local estável, o processo é repetido com o corte atualizado. Caso $P'(M)$ seja inviável ou a solução local não anule o termo de penalidade, Önal deduz que não mais existe solução satisfazendo o novo conjunto de restrições e as folgas complementares e conclui que a última solução viável obtida é ε-ótima para PLDNP.

Em [18], Campêlo e Scheimberg mostraram que mesmo sendo encontrada uma solução local de $P'(M)$ que não anule o termo de penalidade, ainda pode haver soluções viáveis para PLDNP com um valor maior da função objetivo, concluindo-se portanto que as conclusões de Önal estão equivocadas. Além disto, os autores mostraram que o algoritmo de Önal não é capaz de lidar com casos onde a solução de PLDNP é ilimitada. Mais ainda, a hipótese de compacidade do conjunto viável rela-

xado, apesar de não ser referida em Önal [44], é necessária como mostram Campêlo e Scheimberg.

2.1.2 Penalização do *gap* de dualidade

Em [56], White e Anandalingam desenvolvem um algoritmo para resolver PLDNP globalmente utilizando-se a formulação equivalente (P_2).

Defina-se primeiramente os poliedros W (conjunto viável relaxado de PLDNP) e U (conjunto de restrições do problema dual do segundo nível).

$$W = \{(x, y) \geq 0 : A_1x + A_2y \leq a\}, \quad U = \{u \geq 0 : A_2^T u \geq d\}$$

Os autores assumem que as seguintes hipótesis devem ser satisfeitas para uma boa definição do algoritmo:

[A1] Se $x^\star$ é uma solução ótima do líder então existe apenas um ponto pertencente ao $\arg\max\{d^T y : A_2y \leq a - A_1x^\star\}$.

[A2] Os poliedros W e U são não vazios e compactos.

O algoritmo proposto por White e Anandalingam faz uso do seguinte problema penalizado $P(M)$, obtido ao se mover em (P_2) o *gap* de dualidade para a função objetivo do líder, penalizado através de um parâmetro $M \geq 0$:

$$\begin{array}{lll} P(M) & \max & F_M(x, y, u) = c_1^T x + c_2^T y - M\left[u^T(a - A_1x) - d^T y\right] \\ & \text{s.a} & (x, y) \in W, \ u \in U \end{array}$$

Esse algoritmo resolve PLDNP através de uma sequência de problemas $P(M)$, onde o parâmetro M é incrementado ao longo das iterações, diferentemente do algoritmo anterior em que M era considerado implicitamente como um parâmetro dominante na função F_M.

White e Anandalingam utilizam na verdade a seguinte reformulação de $P(M)$, como um problema de maximização de uma função convexa sujeita a um poliedro:

$$P(M) \quad \max_{u \in U} \left\{ \max_{(x,y) \in W} c_1^T x + c_2^T y - M\left[u^T(a - A_1x) - d^T y\right] \right\} = \max_{u \in U} \{\Theta(u, M)\}$$

onde $\Theta(u, M)$ é uma função convexa em $u \in \Re^{m_2}$, para M fixo.

O algoritmo proposto primeiramente determina um ótimo local para o problema $\max\{\Theta(u, M) : u \in U\}$ obtendo um vértice $u \in U$ melhor que seus vértices adjacentes. Na fase seguinte, o algoritmo utiliza uma modificação do algoritmo de Tuy [51], que gera cortes para excluir este ótimo local e aplica testes para verificar se existe um outro vértice com um valor melhor da função objetivo, para o valor atual de M. Se estes cortes tornarem o problema inviável, então tem-se uma solução de $\max\{\Theta(u, M) : u \in U\}$ para um dado valor de M. O procedimento é então reiniciado com o valor de M incrementado. Se esta solução anular o GAP de dualidade então tem-se um ótimo global de PLDNP.

Em [17], Campêlo *et al.* contornam problemas encontrados neste algoritmo. Primeiramente, mostram que a hipótese [A2] é inválida, pois os poliedros W e U não podem ser simultaneamente limitados. Adicionalmente Campêlo *et al.* conseguem os mesmos resultados teóricos, substituindo [A1] e [A2] por uma hipótese mais fraca, qual seja PLDNP tem solução. Um outro problema foi detectado na definição do conjunto de cortes utilizado para excluir um ótimo local e no teste que verifica se o ótimo global já foi atingido para um dado valor de M. Ambos os problemas também foram contornados em Campêlo *et al.* [17], com a redefinição dos cortes. Os autores mostram ainda que a hipótese de compacidade do conjunto viável relaxado W é necessária para uma correta definição do algoritmo.

2.2 Algoritmos *Branch-and-Bound*

O método de *branch-and-bound* é um método geral de otimização global (Horst *et al.* [33, seção 3.7]). Embora seu uso seja mais frequente em programação linear inteira, este método pode ser aplicado aos mais diversos tipos de problemas de programação matemática. O algorimo, em linhas gerais, segue basicamente a seguinte estratégia:

Dado um problema de maximização, considera-se então um problema relaxado que permite a obtenção de um limite superior para o valor ótimo do problema original. Se possível é também determinado um limite inferior. Essas são as etapas de limitação (*bounding*). Se necessário o problema é particionado em subproblemas cujas soluções determinam a solução do problema original. Assim, dá-se o processo de ramificação (*branching*). Em cada subproblema são aplicados testes de eliminação ou poda (*prunning*) com o objetivo de interromper o futuro particionamente deste

subproblema. Os subproblemas que ainda não foram eliminados são armazenados em uma lista (L). Um subproblema de (L) é selecionado e considerado pelo mesmo processo.

O algoritmo *branch-and-bound* pode ser visualizado através de uma árvore de enumeração. Nesta árvore pode ser representado a hierarquia existente entre um subproblema e seus subproblemas descendentes. Uma caracterização para os métodos *branch-and-bound* é feita por Mitten [43], na qual as estratégias de seleção, particionamento (*branching*), limitação (*bounding*) e eliminação (*prunning*) são apresentadas. As regras de divisão, limitação e eliminação são particulares a cada tipo de problema de programação matemática que se pretenda resolver utilizando-se *branch-and-bound*, no entanto a regra de seleção é comum para todos os casos e por isto é apresentada abaixo com mais detalhes.

O processo de seleção determina dentre os nós (subproblemas) ainda não explorados aquele que será explorado na próxima iteração do algoritmo. A seleção pode ser de dois tipos:

1. **a priori** (a ordem de seleção dos nós da árvore é inicialmente determinada), por exemplo: *Depth-First* (testar os subproblemas abaixo de um nó i antes de testar outros subproblemas que estejam no mesmo nível de i) e *Breadth-First* (testar todos os subproblemas de um mesmo nível antes de testar subproblemas do próximo nível).

2. **adaptativas** (o próximo nó é selecionado, a cada iteração, dentre os subproblemas abertos na lista (L)), por exemplo: *Best-First* (escolher sempre o elemento da lista (L) com maior limite superior, ou seja, o subproblema mais promissor), *Best-Estimate* (escolher o elemento de (L) mais provável, segundo algum critério, de apresentar uma melhor solução viável) e *Quick-Improvement* (escolher o elemento de (L) que forneça maior melhora da solução atual).

Os algoritmos apresentados a seguir utilizam o método *branch-and-bound* para resolver globalmente PLDN. Apesar da estratégia global ser comum a todos os algoritmos, eles diferem nas regras de divisão, limitação e eliminação empregadas, mesmo porque se baseiam em diferentes formulações equivalentes de PLDN.

2.2.1 Programa linear {0-1} misto equivalente

Em 1981, Fortuny-Amat e McCarl [28] propuseram uma reformulação do problema (P_1) acrescido das restrições do primeiro nível, que agora será referido por (P_1'), como um problema de programação linear {0-1} mista, onde as restrições de complementaridade são substituídas por um outro conjunto de restrições que envolvem variáveis binárias. Assim origina-se o modelo (PI).

$$
\begin{array}{llll}
(P_1') & \max & c_1^T x + c_2^T y & \\
& \text{s.a} & B_1 x + B_2 y \leq b & \\
& & A_1 x + A_2 y + w = a & \\
& & A_2^T u - v = d & \\
& & x \geq 0, y \geq 0, w \geq 0 & \\
& & u \geq 0, v \geq 0 & \\
& & v^T y = u^T w = 0 & \longrightarrow
\end{array}
\qquad
\begin{array}{llll}
(PI) & \max & c_1^T x + c_2^T y & \\
& \text{s.a} & B_1 x + B_2 y \leq b & \\
& & A_1 x + A_2 y + w = a & \\
& & A_2^T u - v = d & \\
& & x \geq 0, y \geq 0, w \geq 0 & \\
& & u \geq 0, v \geq 0 & \\
& & y \leq M\nu & \\
& & v \leq M(e_1 - \nu) & \\
& & w \leq M\omega & \\
& & u \leq M(e_2 - \omega) & \\
& & \nu \in \{0,1\}^{n_2}, \omega \in \{0,1\}^{m_2} &
\end{array}
$$

onde $e_1^T = (1, 1, \ldots, 1) \in \Re^{n_2}$, $e_2^T = (1, 1, \ldots, 1) \in \Re^{m_2}$, $\{0,1\}^p$ representa o conjunto dos vetores p-dimensionais com componentes 0-1 e M uma constante suficientemente grande.

Admitindo-se que PLDN possui solução ótima, é fácil ver que o problema (PI) possui a mesma solução ótima, pois as funções objetivo são idênticas e um ponto $(x, y, w, u, v, \nu, \omega)$ apenas será viável para (PI) quando $v^T y = u^T w = 0$.

Considerando-se $z^\star = (x^\star, y^\star, w^\star, u^\star, v^\star)$ tal que $(x^\star, y^\star)$ é solução ótima de PLDN, os autores assumem o valor de M como sendo $M \geq \|z^\star\|_\infty$, ou seja, $M \geq \max\{|z_j{}^\star| : j = 1, 2, \ldots, n_1 + n_2\}$. Uma outra alternativa para determinar o valor de M, seria determinar o hipercubo $H(M)$ que contenha todos os vértices do conjunto viável relaxado W de PLDN, ou seja, determinar M tal que $W \subseteq H(M)$.

A resolução de (PI) é realizada com o uso de ferramentas usuais de programação linear inteira. No entanto, note que para valores grandes de n_2 e m_2, a resolução do mesmo torna-se extremanente difícil do ponto de vista computacional.

2.2.2 Fixação de variáveis complementares

Em 1990, Bard e Moore [6] desenvolveram um algoritmo tipo *branch-and-bound* para resolver globalmente um problema de programação linear-quadrática em dois níveis dado pela seguinte formulação:

$$
\begin{array}{lll}
\text{(PLQDN)} & \max\limits_{x,y} & F(x,y) = {c_1}^T x + {c_2}^T y \\
& \text{s.a} & Dx \geq d \\
& & x \geq 0,\ y \text{ solução de:} \\
& & \max\limits_{y}\ f(x,y) = {c_3}^T y + x^T Q_1 y + \frac{1}{2} y^T Q_2 y \\
& & \text{s.a}\ Ax + By \geq b \\
& & \quad Ey \geq e \\
& & \quad y \geq 0
\end{array}
$$

onde c_1, c_2, c_3, d, b, e são vetores, A, B, D, E matrizes de dimensões apropriadas e Q_1, Q_2 matrizes simétricas semidefinidas negativas de dimensões apropriadas.

Os autores assumem as hipóteses de compacidade do conjunto viável relaxado e que o problema do segundo nível fornece resposta única para cada $x \geq 0$. O problema do seguidor é então substituído por suas condições de KKT gerando-se um problema (P') e uma estratégia de *branch-and-bound* é aplicada, resolvendo-se em cada nó da árvore de enumeração o problema (P') sem as restrições de complementaridade e com algumas variáveis de folga fixas a zero.

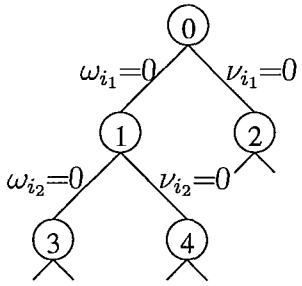
O algoritmo proposto por Bard e Moore [6] pode ser aplicado ao PLDN como pode ser visto em Shimizu *et al.* [48, seção 16.3.2], assumindo-se as mesmas hipóteses de compacidade e resposta única do seguidor.

Como visto anteriormente, o problema (P_1') é obtido ao substituir-se o problema do seguidor em PLDN por suas condições de KKT. O algoritmo proposto trabalha com esta formulação (P_1'), equivalente a PLDN, da seguinte forma:

Como já mencionado anteriormente, uma solução (x, y, w, u, v) é viável para o problema (P_1'), e consequentemente viável para PLDN, quando satisfaz a restrição $v^T y = u^T w = 0$, juntamente com as restrições que definem o conjunto viável relaxado de PLDN. Defina os vetores $\nu \in \Re^{m_2+n_2}$ e $\omega \in \Re^{m_2+n_2}$, como $\nu = (u, v)$ e $\omega = (w, y)$. A restrição $v^T y = u^T w = 0$ pode então ser escrita como $\nu^T \omega = 0$.

A idéia básica do algoritmo é enumerar implicitamente todas as soluções viáveis

de (P_1'). A estratégia de enumeração pode ser visualizada através da seguinte árvore binária.



onde ω_j e ν_j são as j-ésima variáveis de ω e ν respectivamente para $j \in \{1, 2, \ldots, m_2 + n_2\}$.

Cada nó desta árvore está associado a um subproblema do tipo (P_1') excluindo-se a restrição de complementaridade e onde algumas variáveis ω_i e ν_i são fixas a zero. Cada subproblema é identicamente igual ao subproblema do seu pai, acrescido de uma restrição $\omega_i = 0$ ou $\nu_i = 0$ para algum $i \in \{1, 2, \ldots, m_2 + n_2\}$.

Um limite superior para cada nó é dado pela solução de seu subproblema associado. Se este limite superior for menor ou igual a um limite inferior (melhor solução viável conhecida até o momento) ou se o conjunto viável deste subproblema for vazio ou se a solução deste subproblema for viável para (P_1'), este nó não mais necessita ser particionado.

Admitindo-se que o limite superior em um dado subproblema seja maior que o limite inferior então dois subproblema são gerados, a partir deste. O primeiro acrescido da restrição $\omega_k = 0$ e o segundo acrescido da restrição $\nu_k = 0$, onde $k \in \arg\max\{\nu_k \omega_k : k \in \{1, 2, \ldots, m_2 + n_2\}\}$.

O algoritmo claramente termina em um número finito de passos, pois a altura da árvore é limitada a $m_2 + n_2$. Note que a hipótese de compacidade do conjunto viável relaxado é essencial para a boa definição do algoritmo, pois o mesmo se baseia fortemente na solução do problema relaxado. A ilimitação do valor ótimo do problema relaxado comprometeria um dos testes de limitação do algoritmo.

2.2.3 Exploração da estrutura não convexa separável

Em [53], Tuy e Ghannadam propuseram um algoritmo tipo *branch-and-bound* para resolver globalmente PLDN. Os autores exploraram a separibilidade da estrutura não

convexa de um problema equivalente a PLDN, que é um problema de programação reversa convexa (PRC), estudado por Tuy [52]. O problema (PCR) é definido a seguir.

Seja $W_2(x) = \{y \in \Re^{n_2} : A_2 y \leq (a - A_1 x), y \geq 0\}$ o conjunto viável do problema primal do seguidor dado por: $\phi(x) = \max\{d^T y : y \in W_2(x)\}$. Se $\phi(x)$ é finito, pela teoria da dualidade, temos que: $\phi(x) = \min\{(a - A_1 x)^T u : A_2^T u \geq d, u \geq 0\}$.

Os autores consideram em sua formulação o problema $\phi_K(x)$, no qual todas as variáveis do problema $\phi(x)$ são limitadas por um parâmetro $K > 0$. Este problema é dado por: $\phi_K(x) = \min\{(a - A_1 x)^T u : A_2^T u \geq d, 0 \leq u_i \leq K, i = 1, 2, \ldots, m_2\}$.

A função $-\phi_K(x)$ é uma função convexa, pois é o máximo de uma família de funções lineares [46, Teorema 5.5]. Portanto a seguinte propriedade, que será usada mais adiante, é válida [46, Teorema 4.3]:

$$-\phi_K\left(\sum_{i=1}^{k+1} \lambda_i x^i\right) \leq -\sum_{i=1}^{k+1} \lambda_i \phi_K(x^i), \quad \sum_{i=1}^{k+1} \lambda_i = 1, \quad \lambda_i \geq 0 \quad \forall i \in \{1, 2, \ldots, k+1\}$$

A partir da definição de $\phi_K(x)$ é fácil constatar que $d^T y \leq \phi_K(x), \forall y \in W_2(x)$. Visto que em um ponto $(\bar{x}, \bar{y})$ viável para PLDN tem-se $d^T \bar{y} = \phi(\bar{x})$, então o problema (PCR), equivalente a PLDN, pode ser escrito como:

$$\begin{array}{lll} \text{(PCR)} & \max & {c_1}^T x + {c_2}^T y \\ & \text{s.a} & (x, y) \in W \\ & & d^T y - \phi_K(x) \geq 0 \end{array}$$

A restrição $d^T y - \phi_K(x) \geq 0$ é chamada de restrição convexa reversa. Perceba que esta restrição é equivalente a $d^T y - \phi_K(x) = 0$, pois por definição $d^T y - \phi_K(x) \leq 0$.

Os autores desenvolvem então um algoritmo *branch-and-bound*, cuja estratégia básica consiste, inicialmente, em gerar um simplex S_0 (com $k+1$ vértices) no espaço de dimensão $\Re^k$ que contenha a imagem do poliedro W segundo o mapeamento $(x, y) \mapsto Ex$, ou seja, $S_0 \supseteq \{Ex \in \Re^k : (x, y) \in W\}$ (a matriz E é definida a seguir) e particionar este simplex em vários subsimplex S (a construção de S_0 é discutida em [53]).

O valor de k é obtido da seguinte forma:

Tome k=posto(A_1) e $E \in \Re^{k \times n_1}$ uma submatriz de A_1 formada por suas linhas linearmente independentes. Particione-se $E = [B \;\; N]$, onde $B \in \Re^{k \times k}$ é uma base de E, e defina-se $Z^T = [(B^{-1})^T \;\; 0] \in \Re^{k \times n_1}$. Tuy e Ghannadam [53] mostram que

(PCR) é equivalente a:

$$\begin{array}{ll} \max & {c_1}^T x + {c_2}^T y \\ \text{s.a} & (x,y) \in W \\ & Ex = t \\ & d^T y - \phi_K(Zt) \geq 0 \end{array}$$

onde $t \in S_0$.

Cada nó está associado a um subsimplex S proveniente do particionamento do subsimplex do nó pai. Seja então um subsimplex $S = [s^1, s^2, \ldots, s^j, \ldots, s^{k+1}]$ onde s^j representa um vértice qualquer. Tomando-se t como uma combinação convexa destes vértices ($t = \sum_{i=1}^{k+1} \lambda_i s^i$, $\sum_{i=1}^{k+1} \lambda_i = 1 \ \forall i$), em cada nó é calculado um limite superior para (PCR) resolvendo-se o seguinte problema F_2:

$$F_1 \quad \begin{array}{ll} \max & {c_1}^T x + {c_2}^T y \\ \text{s.a} & (x,y) \in W \\ & Ex = \sum_{i=1}^{k+1} \lambda_i s^i \\ & \sum_{i=1}^{k+1} \lambda_i = 1, \ \ \lambda_i \geq 0 \ \forall i \\ & d^T y - \phi_K(\sum_{i=1}^{k+1} \lambda_i Z s^i) \geq 0 \end{array} \quad \longrightarrow \quad F_2 \quad \begin{array}{ll} \max & {c_1}^T x + {c_2}^T y \\ \text{s.a} & (x,y) \in W \\ & Ex = \sum_{i=1}^{k+1} \lambda_i s^i \\ & \sum_{i=1}^{k+1} \lambda_i = 1, \ \ \lambda_i \geq 0 \ \forall i \\ & d^T y - \sum_{i=1}^{k+1} \lambda_i \phi_K(Z s^i) \geq 0 \end{array}$$

Perceba que a restrição $d^T y - \phi_K(\sum_{i=1}^{k+1} \lambda_i Z s^i) \geq 0$ em F_1 impossibilita a sua resolução como um problema linear usual, pois continua sendo hierárquico em λ. No entanto, $d^T y - \sum_{i=1}^{k+1} \lambda_i \phi_K(Z s^i) \geq d^T y - \phi_K(\sum_{i=1}^{k+1} \lambda_i Z s^i) \geq 0$, devido a convexidade de $-\phi_K(x)$. Conclui-se então que F_2 é uma relaxação de F_1. Os autores assumem ainda que o problema relaxado possua solução e que o conjunto $\{Ex \in \Re^k : (x,y) \in W\}$ seja compacto.

O algoritmo resolve em cada nó o problema F_2. Se o problema for inviável ou seu valor ótimo for menor ou igual ao limite inferior atual, o subsimplex não mais precisa ser particionado. Se em determinado subsimplex a solução $\left(\bar{x}, \bar{y}, \bar{\lambda}\right)$ do problema F_2 associado tiver alguma componente $\bar{\lambda}_i$ do vetor $\bar{\lambda}$ igual a 1, significa que $d^T \bar{y} - \sum_{i=1}^{k+1} \bar{\lambda}_i \phi_K(Z s^i) = d^T \bar{y} - \phi_K(\sum_{i=1}^{k+1} \bar{\lambda}_i Z s^i)$, ou seja, o ponto encontrado é um vértice de W viável para PLDN.

O processo de particionamento de cada subsimplex S pode ser realizado através de um ponto interior deste subsimplex, o que gerará $k+1$ subsimplex, ou através de um ponto pertencente a uma aresta deste subsimplex o que gerará apenas 2 subsimplex. Maiores detalhes sobre o processo de particionamento são encontrados na referência original [53].

2.3 Complementaridade linear paramétrica

Em muitos dos algoritmos descritos anteriormente, uma característica comum observada era que o problema do seguidor era substituído por suas KKT. Esta substituição originava um problema (P_1) equivalente a PLDNP. Este problema poderia ser visto como um problema de complementaridade linear (PCL), onde o objetivo fosse encontrar não apenas um ponto complementar, mas dentre estes, encontrar aquele que maximizasse a função objetivo do líder.

Em 1992, Júdice e Faustino [37] desenvolveram um algoritmo para se determinar ε-soluções globais de PLDNP, resolvendo-se uma sequência de problemas de complementaridade linear (PCL). Júdice e Faustino acrescentam ao conjunto de restrições de (P_1) a restrição $c_1^T x + c_2^T y \geq \lambda$, que descarta todos os pontos complementares que possuem, na função objetivo do líder, um valor menor que λ. O problema $PCL(\lambda)$ pode ser formulado como:

$$PCL(\lambda) \quad \begin{pmatrix} w \\ v \\ w_0 \end{pmatrix} = \begin{pmatrix} a \\ -d \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \lambda + \begin{pmatrix} 0 & -A_2 & -A_1 \\ A_2^T & 0 & 0 \\ 0 & c_2^T & c_1^T \end{pmatrix} \begin{pmatrix} u \\ y \\ x \end{pmatrix}$$

$$x \geq 0, y \geq 0, w \geq 0, w_0 \geq 0, u \geq 0, v \geq 0, \quad v^T y = u^T w = 0$$

O algoritmo resolve uma sequência de $PCL(\lambda)$, onde o parâmetro λ é incrementado ao longo das iterações. Tome então (x^k, y^k) uma solução do problema $PCL(\lambda_k)$ em uma dada iteração k, então na iteração $k+1$ o valor de λ_{k+1} será $c_1^T x^k + c_2^T y^k + \gamma |c_1^T x^k + c_2^T y^k|$, onde γ é um número positivo pequeno. O algoritmo é descrito a seguir:

Passo 0: Faça k=0;

Passo 1: Resolva $PCL(\lambda)$ sem considerar a restrição $c_1^T x + c_2^T y \geq \lambda$. Se o problema não tem solução vá para o **Passo 3**, caso contrário seja (x^0, y^0) a solução encontrada, faça k=k+1 e $\lambda_k = c_1^T x^{k-1} + c_2^T y^{k-1} + \gamma |c_1^T x^{k-1} + c_2^T y^{k-1}|$.

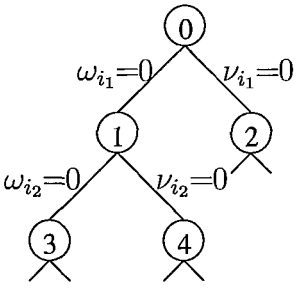
Passo 2: Resolva $PCL(\lambda_k)$. Se o problema não tem solução vá para o Passo 3, caso contrário, seja (x^k, y^k) a solução obtida, faça k=k+1 e $\lambda_k = c_1^T x^{k-1} + c_2^T y^{k-1} + \gamma |c_1^T x^{k-1} + c_2^T y^{k-1}|$. Repita o **Passo 2**.

Passo 3: Se k=0 então PLDNP é inviável, caso contrário (x^k, y^k) é um ε-ótimo global para PLDNP.

Os autores não exigem hipótese de compacidade do conjunto viável relaxado de PLDNP, no entanto pode ser inferido do algoritmo que se PLDNP for ilimitado, o algoritmo não termina. Isto ocorre porque após se aplicar um corte com um valor maior que o da última solução complementar encontrada, o algoritmo sempre encontra uma solução de $PCL(\lambda_k)$.

Um outro aspecto interessante é que o valor de ε dado por $\varepsilon = \gamma|c_1^T x^k + c_2^T y^k|$ é crescente. O parâmetro γ pode ser visto como um percentual máximo do valor ótimo global que define a tolerância com relação a ε-solução encontrada pelo algoritmo. Perceba que para valores grandes de γ a imprecisão do algoritmo será grande e para valores muito pequenos a quantidade de problemas $PCL(\lambda_k)$ pode ser grande, comprometendo a eficiência do algoritmo. É necessário assim um método eficiente de resolução de $PCL(\lambda_k)$ que é apresentado a seguir.

Os autores propuseram um método enumerativo para resolver $PCL(\lambda_k)$. Defina-se primeiramente os vetores $\nu \in \Re^{m_2+n_2}$ e $\omega \in \Re^{m_2+n_2}$, como $\nu = (u, v)$ e $\omega = (w, y)$. A restrição $v^T y = u^T w = 0$ então ser escrita como $\nu^T \omega = 0$. A idéia básica do algoritmo é enumerar implicitamente as soluções viáveis de $PCL(\lambda_k)$. A estratégia de enumeração pode ser visualizada através da seguinte árvore binária.



onde ω_j e ν_j são as j-ésima variáveis de ω e ν respectivamente para $j \in \{1, 2, \ldots, m_2 + n_2\}$.

Ao nó inicial está associado o problema $PCL(\lambda_k)$. A cada outro nó está associado um subproblema que é identicamente igual ao subproblema do seu pai, acrescido de uma restrição $\omega_i = 0$ ou $\nu_i = 0$ para algum $i \in \{1, 2, \ldots, m_2 + n_2\}$. O subproblema relaxado é definido como $\min \nu_i$ ou $\min \omega_i$ sujeito às restrições do problema pai.

Se em um dado nó o valor de $\min \nu_i$ ou $\min \omega_i$ for nulo, então a restrição $\nu_i = 0$ ou $\omega_i = 0$ é acrescentada em todos os subproblemas descendentes. Se o valor de $\min \nu_i$ ou $\min \omega_i$ for positivo, então este nó não mais necessita ser particionado. O

processo enumerativo termina tão logo uma solução complementar seja encontrada, ou seja, a árvore não precisa ser totalmente enumerada, salvo quando o problema $PCL(\lambda_k)$ não possuir solução. Os autores sugerem ainda o uso de heurísticas para reduzir o número de subproblemas gerados.

Finalmente, vale a pena mencionar que este método enumerativo possui a sua regra de ramificação semelhante aquela utilizada pelo algoritmo *branch-and-bound* proposto por Bard e Moore (seção 2.2.2, pg.26) para a resolução de PLDN.

2.4 Enumeração de pontos extremos

Em 1984, Bialas e Karwan [13] desenvolveram um algoritmo global para a resolução de PLDNP, adimitindo-se a compacidade do conjunto viável relaxado $W = \{(x, y) \geq 0 : A_1x + A_2y \leq a\}$ e resposta única do problema do seguidor para cada $x \geq 0$.

A estratégia utilizada não faz uso de nenhuma formulação equivalente a PLDNP, como mostrado nos algoritmos anteriores. Na verdade, o algoritmo utiliza uma estratégia bem simples baseada na enumeração dos vértices do conjunto viável relaxado W. Defina-se T como o conjunto de vértices de W a analizar e $\bar{T}$ o conjunto de vértices já analizados de W. O algoritmo é definido a seguir:

Passo 1: Faça $i = 1$. Resolva o problema relaxado RPLDNP, obtendo uma solução ótima (x^i, y^i). Faça $T = \{(x^i, y^i)\}$ e $\bar{T} = \emptyset$.

Passo 2: Se (x^i, y^i) for viável para PLDNP **pare**, (x^i, y^i) é um ótimo global. Caso contrário vá para o **Passo 3**.

Passo 3: Seja T^i o subconjunto de vértices (x, y) de W, adjacentes a (x^i, y^i), tais que $c_1^T x + c_2^T y \leq c_1^T x^i + c_2^T y^i$. Faça $\bar{T} = \bar{T} \cup \{(x^i, y^i)\}$ e $T = (T \cup T^i) \setminus \bar{T}$. Vá para o **Passo 4**.

Passo 4: Faça $i = i + 1$ e escolha $(x^i, y^i) \in \arg\max\{c_1^T x + c_2^T y : (x, y) \in T\}$. Vá para o **Passo 2**.

Como pode ser visto o algoritmo apresentado realiza uma enumeração dos vértices de W. Os autores observam que o desempenho do algoritmo aumenta à medida que se diminue a distância relativa entre a solução de RPLDNP e o ótimo de PLDNP.

Capítulo 3

Algoritmos Estudados

Nos capítulos anteriores, o problema de programação linear em dois níveis (PLDN) foi definido. Suas principais propriedades e aplicações foram apresentadas, bem como os principais algoritmos existentes voltados à resolução global de PLDN. Neste capítulo é apresentado em detalhes dois algoritmos desenvolvidos para resolver PLDNP globalmente.

O presente trabalho tem como objetivo principal o estudo computacional do recente algoritmo global, desenvolvido por Campêlo [16], para a resolução de PLDNP. Este estudo baseou-se na comparação entre este algoritmo e um dos algoritmos mais eficientes conhecidos na literatura desenvolvido por Hansen *et al.* [29].

Este estudo computacional, como será apresentado adiante, teve uma importância fundamental, pois através do mesmo foram indroduzidas modificações tanto neste algoritmo como no algoritmo de Hansen *et al.*. A escolha deste último algoritmo como base para o estudo de comparação realizado neste trabalho foi baseada nos resultados computacionais obtidos da literatura, para os algoritmos apresentados no capítulo 2. A seguir são enumeradas algumas das principais conclusões, referente às comparações realizadas, obtidas da literatura para estes algoritmos.

1. White e Anandalingam realizaram em [56] um estudo computacional comparando o algoritmo de penalidades desenvolvido por eles (seção 2.1.2, pg. 22), com os algoritmos de Bard e Moore [6] apresentado na seção 2.2.2 (pg.26) e Bialas e Karwan [13] apresentado na seção 2.4 (pg. 32). Neste estudo o algoritmo de White e Anandalingam superou facilmente o algoritmo enumerativo de Bialas e Karwan e mostrou-se similar ao algoritmo de Bard e Moore para problemas de pequena dimensão, no entanto para problemas de dimensões

maiores o algoritmo de Bard e Moore superou o algoritmo de White e Anandalingam.

2. Em [37] Júdice e Faustino compararam o seu algoritmo de complementaridade linear paramétrica apresentado na seção 2.3 (pg. 30) com o algoritmo de Bard e Moore [6]. Neste estudo o algoritmo de Bard e Moore mostrou-se semelhante, e algumas vezes bem melhor, que o de Júdice e Faustino para problemas de pequena dimensão, no entanto, este último mostrou-se significantemente mais eficiente que o outro para problemas de dimensões maiores ($n_1 + n_2 \geq 50$).
3. O maior número de testes existentes na literatura foi apresentada por Hansen *et al.* [29]. O algoritmo de Hansen *et al.* foi comparado ao algoritmo de Bard e Moore [6], utilizando-se o código fornecido pelos mesmos. No estudo realizado por Hansen *et al.* o seu algoritmo superou o último em todos os testes realizados, com uma diferença significativa. Além disto, os resultados obtidos por Hansen *et al.* mostraram-se similar aos resultados obtidos por Júdice e Faustino [37] que testou problemas de dimensões e densidades similares.

Baseando-se nestas observações, pode-se concluir que o método desenvolvido por Hansen *et al.* se encontra entre os mais eficientes da literatura. No estudo comparativo realizado neste trabalho o código do algoritmo de Campêlo [16] foi fornecido pelo autor, no entanto, o algortimo de Hansen *et al.* [29] necessitou ser implementado, pois os autores não tem à disposição seu código ou mesmo seus problemas testes. A seguir os dois algoritmos são apresentados em detalhes.

3.1 Algoritmo de Campêlo

O algoritmo descrito nesta seção, desenvolvido por Campêlo [16], encontra ε-soluções globais do PLDNP, através da busca de pontos extremos do conjunto viável relaxado W. Uma de suas principais características é que este algoritmo, comparativamente aos outros algoritmos existentes na literatura para a resolução de PLDN, não necessita da hipótese de compacidade do conjunto viável relaxado W.

Esta característica permite que o algoritmo identifique os casos de inviabilidade e ilimitação. Em particular, o algoritmo é capaz de solucionar problemas onde a solução ótima de PLDNP é limitada, mas o problema relaxado é ilimitado.

O algoritmo de Campêlo trabalha com a formulação (P_1), equivalente a PLDNP (pg.20), onde o problema do seguidor é substituído por suas condições de otimalidade de Karush-Kuhn-Tucker (KKT). Esta formulação foi utilizada anteriormente, como apresentado no capítulo anterior, por White e Anandalingam [56], Önal [44], Bard e Moore [6] e Júdice e Faustino [37]. Esta formulação elimina o problema do seguidor, mas o conjunto de restrições de (P_1) continua sendo não convexo devido às restrições de complementaridade.

Sendo este conjunto não convexo, Campêlo penaliza, através de um parâmetro $M \geq 0$, as restrições de complementaridade na função objetivo do líder, obtendo-se o problema abaixo, considerado anteriormente por Önal [44] como apresentado no capítulo anterior (pg.20).

$$
\begin{aligned}
P(M) \quad \max \quad & c_1^T x + c_2^T y - M(v^T y + u^T w) && (3.1)\\
\text{s.a} \quad & A_1 x + A_2 y + w = a && (3.2)\\
& x \geq 0, y \geq 0, w \geq 0 && (3.3)\\
& A_2^T u - v = d && (3.4)\\
& u \geq 0, v \geq 0 && (3.5)
\end{aligned}
$$

Fazendo-se uso do conjunto primal e dual do seguidor definidos respectivamente como $Z = \{z \in \Re^n : Az = a, z^T = (x^T, y^T, w^T) \geq 0\}$ e $S = \{s \in \Re^n : Ds = d, s^T = (0, v^T, u^T) \geq 0\}$, onde $A = [A_1\ A_2\ I_{m_2}] \in \Re^{m_2 \times n}$ e $D = [0\ -I_{n_2}\ A_2^T] \in \Re^{n_2 \times n}$ são matrizes bloco, $c^T = (c_1^T, c_2^T, 0) \in \Re^n$ um vetor, $n = n_1 + n_2 + m_2$, I_k uma matriz identidade de ordem k e 0 uma matriz nula de dimensões apropriadas, Campêlo reescreve os problemas (P_1) e $P(M)$ como:

$$
\begin{array}{llll}
(P_1) & \max & F(z,s) = c^T z & \\
& \text{s.t.} & z \in Z, s \in S, & \\
& & s^T z = 0, &
\end{array}
\qquad
\begin{array}{llll}
P(M) & \max & F_M(z,s) = c^T z - M s^T z & \\
& \text{s.t.} & z \in Z, s \in S & \\
& & &
\end{array}
$$

Note que o problema $P(M)$ formulado desta maneira pode ser visto como uma família de problemas bilineares (pg.8) parametrizados por M.

As relações entre as soluções ótimas de (P_1) e $P(M)$, bem como os casos de ilimitação e inviabilidade foram estudados por Campêlo [16], que deduziu o seguinte teorema:

Teorema 3.1.1 *[16, pg.77] Exatamente um dos seguintes casos acontece:*

(1) os problemas (P_1) *e* $P(M)$ *são inviáveis para todo* $M \geq 0$*;*

(2) os problemas (P_1) *e* $P(M)$ *são ilimitados para todo* $M \geq 0$*;*

(3) os problemas (P_1) *e* $P(M)$ *têm o mesmo conjunto (não vazio) de soluções globais para todo* $M > M^\star$*, para algum* $M^\star \geq 0$*.*

Enunciado este teorema que relaciona as soluções de (P_1) e $P(M)$, o seguinte teorema é enunciado em [16, Teorema 3.2.2] relacionando as soluções do problema penalizado $P(M)$ com as soluções do problema original PLDNP.

Teorema 3.1.2 *[16, pg.77] Os problemas PLDNP e* $P(M)$ *são ambos inviáveis ou ilimitados para todo* $M \geq 0$*, ou então eles têm solução para algum* $M_0 \geq 0$*. Neste último caso,* $z^\star$ *é uma solução global de PLDNP se, e somente se, existe* $s^\star$ *tal que* $(z^\star, s^\star)$ *é solução global de* $P(M)$ *para todo* $M > M^\star$*, para algum* $M^\star \geq M_0$*.*

Como citado anteriormente, os problemas PLDNP e (P_1) possuem suas regiões viáveis não convexas e o problema $P(M)$ possui sua função objetivo $F_M(z, s)$ não convexa, o que possibilita a existência de ótimos locais.

De modo a se estabelecer relações entre os ótimos locais de PLDNP, (P_1) e $P(M)$ Campêlo introduz o conceito de ponto de equilíbrio do problema penalizado $P(M)$ [16, pg.82]:

Definição 3.1.1 *Um ponto* $(\bar{z}, \bar{s})$ *é um* **ponto de equilíbrio** *do problema penalizado* $P(M)$ *se existe* $\bar{M} \geq 0$ *tal que, para cada* $M \geq \bar{M}$*, verifica-se*

$$\max_{s \in S} F_M(\bar{z}, s) = F_M(\bar{z}, \bar{s}) = \max_{z \in Z} F_M(z, \bar{s}), \tag{3.6}$$

onde $F_M(z, s) = c^T z - M s^T z$*.*

Campêlo mostra que o ponto de equilíbrio $(\bar{z}, \bar{s})$ satisfaz $\bar{z}^T \bar{s} = 0$ [16, Proposição 3.3.1] e é uma solução local de (P_1) [16, Teorema 3.3.2]. O autor desenvolve [16, pg.107] o seguinte algoritmo para se encontrar um ponto de equlíbrio:

Algoritmo 1 (Ponto de equilíbrio de $P(M)$)

Passo 0 Sejam $Z \times S \neq \emptyset$ e $z^0 \in Z$.

Passo 1 Resolva $\max_{s \in S} F_M(z^0, s)$ pelo método simplex, obtendo uma solução $\bar{s}$.

Passo 2 Resolva $\max_{z \in Z} F_M(z, \bar{s})$, pelo método simplex *big-M*. Obtenha uma solução $\bar{z}$ ou verifique que o problema é ilimitado. No primeiro caso, $(\bar{z}, \bar{s})$ é um ponto de equilíbrio; no segundo, conclua que $P(M)$ é ilimitado para todo $M \geq 0$.

O algoritmo proposto termina com uma das seguintes possibilidades [16, pg.108]:

(i) encontra um ponto de equilíbrio do problema penalizado $P(M)$ e, equivalentemente, uma solução local de (P_1), ou

(ii) verifica que $P(M)$ é ilimitado para todo $M \geq 0$ e, equivalentemente, que (P_1) e PLDNP são ilimitados.

Analogamente ao algoritmo 1, um outro procedimento é definido pelo autor para se determinar um ponto de equilíbrio a partir de um ponto $s^0 \in S$ [16, pg.108].

Perceba que o ponto de equilíbrio $(\bar{z}, \bar{s})$ é determinado em apenas dois problema lineares. Além disso é interessante notar que entre todos os pontos $z \in Z$ satisfazendo $\bar{s}^T z = 0$, o ponto $\bar{z}$ é aquele que fornece o maior valor da função objetivo do líder, em outras palavras, $\bar{z}$ é o melhor ponto viável para PLDNP na face $\{z \in Z : \bar{s}^T z = 0\}$.

O algoritmo global proposto por Campêlo para a resolução de PLDNP utiliza em sua iteração inicial o algoritmo 1, para se determinar um ponto de equilíbrio de $P(M)$ e consequentemente uma solução local de (P_1), ou verificar que PLDNP é inviável ou ilimitado. Após determinado este ponto de equilíbrio inicial (z^*, s^*) o algoritmo global passa a trabalhar com os seguintes problemas:

$$(\tilde{P}_1) \quad \begin{array}{ll} \max & F(z,s) = c^T z \\ \text{s.a} & z \in \tilde{Z}, s \in S \\ & s^T z = 0 \end{array} \qquad \tilde{P}(M) \quad \begin{array}{ll} \max & F_M(z,s) = c^T z - M s^T z \\ \text{s.a} & z \in \tilde{Z}, s \in S \end{array}$$

onde $F^* = F(z^*, s^*)$ e $\tilde{Z} = \{z \in \Re^n : c^T z \geq F^* + \varepsilon, Az = a, z \geq 0\} \subseteq Z$, com $\varepsilon > 0$.

O conjunto $\tilde{Z}$ é exatamente igual ao conjunto Z acrescido de um corte linear, ou seja, $\tilde{Z} = Z \cap \{z \in \Re^n : c^T z \geq F^* + \varepsilon\}$, onde F^* é o valor da função objetivo do líder no melhor ponto viável para PLDNP conhecido até o momento.

Perceba que este corte linear isola a melhor solução viável conhecida até o momento. De fato, ao se tentar resolver $\tilde{P}(M)$, o ótimo global, se existir, estará a

uma distância de pelo menos ε da melhor solução viável de PLDNP conhecida até o momento. O uso deste tipo de corte pode ser observado nos algoritmos de Júdice e Faustino [37] e Önal [44].

A introdução deste corte linear no conjunto Z é equivalente a se adicionar a restrição $c^T z \geq F^* + \varepsilon$ no primeiro nível de PLDNP, ou seja, o problema $(\tilde{P}_1)$ é equivalente a PLDN. A equivalência global entre os problemas $(\tilde{P}_1)$ e $\tilde{P}(M)$ se mantém. Entretanto, as relações entre ótimos locais destes problemas são diferentes daquelas obtidas para (P_1) e $P(M)$. Para caracterizar estas relações Campêlo e Scheimberg [15] consideram a seguinte definição de ponto de equilíbrio de $\tilde{P}(M)$, similar a definição 3.1.1:

Definição 3.1.2 *Um ponto $(\bar{z}, \bar{s})$ é um* **ponto de equilíbrio** *do problema penalizado $\tilde{P}(M)$ se existe $\bar{M} \geq 0$ tal que, para cada $M \geq \bar{M}$, verifica-se*

$$\max_{s \in S} F_M(\bar{z}, s) = F_M(\bar{z}, \bar{s}) = \max_{z \in \bar{Z}} F_M(z, \bar{s}), \tag{3.7}$$

onde $F_M(z, s) = c^T z - M s^T z$.

O ponto de equilíbrio do problema $\tilde{P}(M)$, ao contrário do anterior, pode **não** satisfazer $\bar{z}^T \bar{s} = 0$ e portanto não ser uma solução local do problema $(\tilde{P}_1)$. No entanto, como pode ser visto em [16, Teorema 3.4.4], se o ponto de equilíbrio de $\tilde{P}(M)$ satisfizer a complementaridade o mesmo será uma solução local de $(\tilde{P}_1)$.

Campêlo desenvolve o seguinte algoritmo [16, pg.109] para se encontrar um ponto de equlíbrio de $\tilde{P}(M)$ partindo-se de uma solução viável $s^0 \in S$. Um procedimento similar pode ser definido partindo-se de um ponto $z^0 \in Z$.

Algoritmo 2 (Ponto de equilíbrio de $\tilde{P}(M)$)

Passo 0 Sejam $Z \times S \neq \emptyset$ e $s^0 \in S$. Faça $i = 0$.

Passo 1 Resolva $\tilde{P}(M; s^i)$ pelo método simplex *big-M*. Obtenha uma solução z^i ou verifique que o problema é ilimitado. No segundo caso, conclua que $\tilde{P}(M)$ é ilimitado para todo $M \geq 0$.

Passo 2 Resolva $\tilde{P}(M; z^i)$, obtendo uma solução s^{i+1}. Se $F_M(z^i, s^i) = F_M(z^i, s^{i+1})$, pare: (z^i, s^i) é um ponto de equilíbrio.

Passo 3 Resolva $\tilde{P}(M; s^{i+1})$ pelo método simplex *big-M*. Obtenha uma solução z^{i+1} ou verifique que o problema é ilimitado. No segundo caso, conclua que $\tilde{P}(M)$ é ilimitado para todo $M \geq 0$. No primeiro, se $F_M(z^i, s^{i+1}) = F_M(z^{i+1}, s^{i+1})$, pare: (z^i, s^{i+1}) é um ponto de equilíbrio; senão, faça $i = i+1$ e volte ao passo 2.

O algoritmo proposto resolve uma seqüência finita de problemas lineares e termina em um número finito de iterações [16, Proposição 4.1.3]. Além disso o mesmo encontra um ponto de equilíbrio de $\tilde{P}(M)$ ou verifica que $\tilde{P}(M)$ é ilimitado para todo $M \geq 0$ [16, Proposição 4.1.4].

O algoritmo global proposto por Campêlo para resolver PLDNP, gera uma seqüência $\{(z^i, s^i)\} \in \tilde{Z}_v \times S_v$ de pontos viáveis para PLDNP, associada a uma seqüência crescente $\{c^T z^i\}$ de valores na função objetivo do líder. Esta seqüência de pontos é obtida através do algoritmo proposto para se determinar pontos de equilíbrio de $\tilde{P}(M)$. No entanto, como apresentado anteriormente, os mesmos podem não ser viáveis para PLDNP ($z^{i^T} s^i > 0$). Sendo assim um método alternativo deve ser usado para se determinar um ponto $(\bar{z}, \bar{s}) \in \{z \in \tilde{Z}, s \in S, z^T s = 0\}$.

O problema de complementaridade linear (PCL) a ser resolvido é então abordado através do seguinte problema de programação bilinear:

$$\begin{array}{lll} \text{(PCL)} & \min & s^T z \\ & \text{s.a} & (z, s) \in \tilde{Z} \times S \end{array}$$

Note que o problema (PCL) não pode ser ilimitado, pois $z^T s \geq 0, \forall (z, s) \in \tilde{Z} \times S$. Se a solução ótima $(\bar{z}, \bar{s})$ de (PCL) satisfizer $\bar{z}^T \bar{s} = 0$ então esta solução será viável para $(\tilde{P}_1)$ e $\bar{z}$ consequentemente viável para PLDNP. Caso contrário, se a solução ótima $(\bar{z}, \bar{s})$ de (PCL) for $\bar{z}^T \bar{s} > 0$ ou (PCL) for inviável, então $(\tilde{P}_1)$ é inviável, ou seja, não existe nenhum ponto (z, s) complementar satisfazendo o corte linear $c^T z \geq F^* + \varepsilon$, o que leva a conclusão de que a última solução viável conhecida para PLDNP é uma ε-solução global de PLDNP.

Um fato importante a ser observado é que devido a conexidade do conjunto viável de PLDNP (definição 1.3.1, pg.10), se houver pontos $(z, s) \in \{z \in \tilde{Z}, s \in S, z^T s = 0\}$, pelo menos um destes pontos satisfaz o corte linear $c^T z = F^* + \varepsilon$. Sendo assim

o seguinte problema (PCL'), pode ser considerado ao invés do problema (PCL).

$$\begin{array}{lll} \text{(PCL')} & \min & s^T z \\ & \text{s.a} & (z,s) \in Z' \times S \end{array}$$

onde $Z' = \{z \in \Re^n : c^T z = F^* + \varepsilon, Az = a, z \geq 0\} \subseteq \tilde{Z} \subseteq Z$.

A vantagem em se considerar este problema ao invés de (PCL) está em dois motivos. O primeiro deles é que o espaço de soluções de (PCL') é bem menor que o de (PCL) e segundo é que o conjunto Z' é compacto se as curvas de nível da função $c^T z$ são limitadas sobre Z.

Vários métodos são propostos na literatura para a solução de problemas de programação bilinear, como pode ser visto em algumas referencias citadas anteriormente (pg.8). Campêlo utilizou um método tipo *outer approximation* (Horst e Tuy [34, seção IX.1.5]), para a resolução de (PCL'), por ser o mesmo capaz de lidar com conjuntos ilimitados.

O problema (PCL') pode ser reformulado como o problema de minimização da função $g(z) = \min\{z^T s : s \in S\}$ restrita ao conjunto $\{z \in Z'\}$. Perceba que a função $g(z)$ é côncava, pois é definida pelo mínimo de uma família de funções lineares (Rockafellar [46, teorema 5.5]).

O problema (PCL'), para efeito do algoritmo global de Campêlo, não necessita ser resolvido até a sua otimalidade, visto que é suficiente se encontrar um ponto $(\tilde{z}, \tilde{s})$, tal que $\tilde{z}^T \tilde{s} \leq \bar{z}^T \bar{s}$, onde $(\bar{z}, \bar{s}) \in Z'_v \times S_v$ é um ponto de equilíbrio com $\bar{z}^T \bar{s} \geq 0$. Assim o **Algoritmo 2** (pg.38) pode ser reiniciado a partir de $(\tilde{z}, \tilde{s})$.

Devido à concavidade da função $g(z)$ e a convexidade do cojunto Z', a determinação do ponto $(\tilde{z}, \tilde{s})$ pode ser feita pesquisando-se os vértices do conjunto Z'. Partindo-se então de um ponto de equilíbrio $(\bar{z}, \bar{s}) \in Z'_v \times S_v$, com $\bar{z}^T \bar{s} > 0$, Campêlo trabalha com uma reformulação do problema (PCL') na qual as variáveis básicas z_B de $\bar{z}$ são expressas em função das variáveis não básicas $z \equiv z_N$, onde B é uma base associada a $\bar{z}$, tal que $[B\ N] = \begin{bmatrix} A \\ c^T \end{bmatrix}$. O problema (PCL') é então reescrito como:

$$\begin{array}{lll} \text{(PCL')} & \min & \bar{s}^T \bar{z} - s_B^T Q z + s_N^T z \\ & \text{s.a} & Qz \leq q \\ & & s = (s_B^T, s_N^T)^T \in S \end{array}$$

onde $Q = B^{-1}N$ e $q = \bar{z}_B$.

Trabalhando-se com o problema (PCL') reformulado como a minimização de uma função côncava restrita a um poliedro, uma outra formulação pode ser obtida:

$$\begin{aligned} (\text{PCL}') \quad & \min \quad \tilde{g}(z) \\ & \text{s.a} \quad z \in \mathcal{Z}' = \{z \geq 0 : Q_i^T z \leq q_i \;\; i = 1, 2, \ldots, m_2 + 1\} \end{aligned}$$

onde $\tilde{g}(z) = \min\{\bar{s}^T \bar{z} + z^T U^T s \; : \; s \in S\}$, Q_i^T é a i-ésima linha da matriz Q e $U^T = [-Q^T \; I]$.

Perceba que o ponto $z = 0$ é viável para $\mathcal{Z}'$, ou seja, $(0, \bar{s}) \in \mathcal{Z}' \times S$. Na verdade este ponto $(0, \bar{s})$ representa, no espaço de soluções original, o ponto de equilíbrio $(\bar{z}, \bar{s})$.

O método *outer approximation*, na verdade, trabalha com uma relaxação do problema (PCL') onde o poliedro $\mathcal{Z}'$ é substituído por um poliedro $\mathcal{W}^1$ tal que $\mathcal{Z}'_v \subseteq \mathcal{W}^1_v$, onde $\mathcal{Z}'_v$ e $\mathcal{W}^1_v$ são os conjuntos de vértices dos conjuntos $\mathcal{Z}'$ e $\mathcal{W}^1$ respectivamente. No caso em que $\mathcal{Z}'$ é limitado, $\mathcal{W}^1$ pode ser determinado como sendo $\mathcal{W}^1 = \left\{z \geq 0 : \sum_{i=1}^{n-(m_2+1)} z_i \leq L\right\}$, onde $L = \max\{\sum_{i=1}^{n-(m_2+1)} z_i : z \in \mathcal{Z}'\}$. No caso em que $\mathcal{Z}'$ é ilimitado, além dos vértices de $\mathcal{W}^1$ ainda devem ser determinados seus raios extremos (Horst e Tuy [34, algoritmo IX.4]). O algoritmo *outer approximation* é descrito a seguir.

Algoritmo (*Outer Approximation*)

Passo 0 Construa um politopo (limitado) $\mathcal{W}^1$ contendo todos os vértices de $\mathcal{Z}'$. Calcule o conjunto de vértices $\mathcal{W}^1_v$. Seja $J_1 = \{1, 2, \ldots, m_2 + 1\}$ o conjunto de índices das restrições que definem $\mathcal{Z}'$. Faça $l = 1$.

Passo 1 Escolha $\tilde{z} \in \arg\min\{\tilde{g}(z) : z \in \mathcal{W}^l_v\}$.

Passo 2 Se $Q_i^T \tilde{z} \leq q_i$ para todo $i \in J_l$, pare: $(\tilde{z}, \tilde{s})$ é solução de (PCL'), onde $\tilde{s}$ é a solução que fornece $\tilde{g}(\tilde{z})$, ou seja, $\tilde{s} \in \arg\min\{\bar{s}^T \bar{z} + \tilde{z}^T U^T s : s \in S\}$.

Passo 3 Escolha $j \in \arg\max\{Q_i^T \tilde{z} - q_i : i \in J_l\}$. Faça $\mathcal{W}^{l+1} = \mathcal{W}^l \cap \{z : Q_j^T z \leq q_j\}$. Determine o conjunto de vértices $\mathcal{W}^{l+1}_v$ a partir de $\mathcal{W}^l_v$. Faça $J_{l+1} = J_l \backslash \{j\}$, $l = l + 1$ e volte ao passo 1.

Como citado anteriormente, o problema (PCL') não precisa ser resolvido exatamente (como faz o algoritmo anterior), pois é necessário apenas que se encontre

um ponto $\tilde{z}^T\tilde{s} \leq \bar{z}^T\bar{s}$, ou equivalentemente um ponto $z \in \mathcal{Z}'$ tal que $\tilde{g}(z) < \tilde{g}(0)$. Campêlo propõe então uma modificação deste algoritmo levando-se em consideração estas observações. Estas modificações são apresentadas a seguir.

A primeira delas ocorre no passo 2 do algoritmo. Perceba que não é necessário se calcular $\tilde{z} \in \arg\max\{\tilde{g}(z) : z \in \mathcal{W}_v^l\}$, mas apenas se encontrar um ponto $\tilde{z} \in \mathcal{W}_v^l$ tal que $\tilde{g}(\tilde{z}) < \tilde{g}(0)$. Além disso, devido a concavidade da função $\tilde{g}$, se não existe tal ponto $\tilde{z} \in \mathcal{W}_v^l$ satisfazendo a requerida condição, então não existe $p \in [0, \tilde{z}]$ viável para $\mathcal{Z}'$ tal que $\tilde{g}(p) < \tilde{g}(0)$, o que leva a conclusão de que $(0, \bar{s})$ é solução de (PCL').

A segunda modificação, que na verdade é um melhoramento do algoritmo, determina um ponto $\hat{z} \in Z'$ a partir de um ponto $\tilde{z} \in \mathcal{W}_v^l$. Mais uma vez, devido a concavidade da função $\tilde{g}$, se um ponto $\tilde{z} \in \mathcal{W}_v^l$ satisfaz $\tilde{g}(\tilde{z}) < \tilde{g}(0)$, então pode existir um ponto $\hat{z} \in [0, \tilde{z}]$ viável para $\mathcal{Z}'$ tal que $\tilde{g}(\hat{z}) < \tilde{g}(0)$. Campêlo determina então um passo α tal que $\hat{z} = \alpha\tilde{z}$ e $Q\hat{z} \leq q$. Quando $\tilde{g}(\hat{z}) < \tilde{g}(0)$ então o algoritmo é finalizado. O método *Outer Approximation Adaptado* é apresentado a seguir.

Algoritmo (*Outer Approximation Adaptado*)

Passo 0 Construa um politopo (limitado) $\mathcal{W}^1$ contendo todos os vértices de $\mathcal{Z}'$. Calcule o conjunto de vértices $\mathcal{W}_v^1$. Seja $J_1 = \{1, 2, \ldots, m_2 + 1\}$ o conjunto de índices das restrições que definem $\mathcal{Z}'$. Faça $l = 1$.

Passo 1 Encontre $\tilde{z} \in \mathcal{W}_v^l$ tal que $\tilde{g}(\tilde{z}) < \tilde{g}(0)$. Se não existe tal vértice, pare: retorne $(0, \bar{s}) \in \mathcal{Z}' \times S$ solução de (PCL').

Passo 2 Se $Q_i^T\tilde{z} \leq q_i$ para todo $i \in J_l$, pare: retorne $(\tilde{z}, \tilde{s})$ solução de (PCL'), onde $\tilde{s}$ é a solução que fornece $\tilde{g}(\tilde{z})$, ou seja, $\tilde{s} \in \arg\min\{\bar{s}^T\bar{z} + \tilde{z}^T U^T s : s \in S\}$.

Passo 3 Calcule o passo α conforme descrito em Campêlo [16, pg.121] e faça $\hat{z} = \alpha\tilde{z}$. Determine $\tilde{g}(\hat{z})$, obtendo $\hat{s} \in S$. Se $\tilde{g}(\hat{z}) < \tilde{g}(0)$, pare. Retorne $(\hat{z}, \hat{s})$ solução de (PCL').

Passo 4 Escolha $j \in \arg\max\{Q_i^T\tilde{z} - q_i : i \in J_l\}$. Faça $\mathcal{W}^{l+1} = \mathcal{W}^l \cap \{z : Q_j^T z \leq q_j\}$. Determine o conjunto de vértices $\mathcal{W}_v^{l+1}$ a partir de $\mathcal{W}_v^l$. Faça $J_{l+1} = J_l \setminus \{j\}$, $l = l + 1$ e volte ao passo 1.

A solução obtida é, então, convertida ao espaço original ($\Re^n$) sendo $\tilde{z}_N = \tilde{z}$ e

$\tilde{z}_B = \bar{z}_B - Q\tilde{z}_N$, quando a solução é proveniente dos passos 1 ou 2, e $\tilde{z}_N = \hat{z}$ e $\tilde{z}_B = \bar{z}_B - Q\hat{z}_N$, quando a solução é proveniente do passo 3.

Perceba que o algoritmo termina em no máximo $m_2 + 1$ interações, pois este é o limite de cortes que podem ser acrescentados ao poliedro $\mathcal{W}^1$ equivalentes às restrições em $\mathcal{Z}'$. Apesar de haver poucas iterações, o passo 4 possui complexidade elevada, pois no mesmo é necessário calcular todos os novos vértices que aparecem no novo conjunto $\mathcal{W}_v^{l+1}$ ao se acrescentar ao conjunto $\mathcal{W}_v^l$ a restrição $\{z : Q_j^T z \leq q_j\}$. Para o cálculo destes novos vértices Campêlo adota o algoritmo de Chen *et al.* [21] por o mesmo estar entre os mais eficientes da literatura.

A seguir o algoritmo global, desenvolvido por Campêlo, para se determinar uma ε-solução global de PLDNP é apresentado.

Algoritmo (Algoritmo Global)

Passo 1 Se $S = \emptyset$, vá para o passo 6. Senão encontre $\hat{s} \in S$ e considere $\tilde{Z} = Z$. Faça $k = 0$.

Passo 2 Se $\tilde{Z} = \emptyset$, vá para o passo 6.

Passo 3 Aplique o **Algoritmo 2** (pg.38) de equilíbrio a partir de $\hat{s}$. Se $\tilde{P}(M)$ é ilimitado para todo $M \geq 0$, pare: PLDNP é ilimitado. Do contrário, seja $(\bar{z}, \bar{s})$ o ponto de equilíbrio encontrado.

Passo 4 Se $\bar{s}^T\bar{z} = 0$, então $(\bar{z}, \bar{s})$ é solução viável de PLDNP; faça $(z^\star, s^\star) = (\bar{z}, \bar{s})$ e $F^\star = F(\bar{z}, \bar{s})$; aplique um corte linear, definindo $\tilde{Z} = Z \cap \{z \in \Re^n : c^T z \geq F^\star + \varepsilon\}$, com $\varepsilon > 0$; faça $\hat{s} = \bar{s}$ e $k = k + 1$; volte ao passo 2.

Passo 5 Se $\bar{s}^T\bar{z} > 0$, aplique o algoritmo *outer approximation adaptado*, obtendo $(\hat{z}, \hat{s})$. Se $\hat{s}^T\hat{z} \geq \bar{s}^T\bar{z}$, vá para o passo 6. Senão, volte ao passo 3.

Passo 6 Pare. Se $k = 0$, PLDN é inviável. Senão, $(x^\star, y^\star)$ é uma ε-solução global de PLDNP.

O algoritmo proposto pára em um número finito de interações [16, Proposição 4.3.2]. Além disso o mesmo encontra uma ε-solução global de PLDNP ou verifica que o problema é ilimitado ou inviável [16, Proposição 4.3.3].

3.2 Algoritmo de Hansen *et al.*

O algoritmo desenvolvido por Hansen *et al.* [29] encontra uma solução global para PLDN, através da busca de pontos extremos do conjunto viável relaxado W. Este algoritmo, ao contrário do algoritmo de Campêlo, assume que o conjunto viável relaxado W e o conjunto viável do seguidor $W(x)$ são conjuntos não vazios e compactos. Logo abaixo a formulação de PLDN é relembrada para facilitar as explicações que se seguem.

$$
\begin{aligned}
\text{(PLDN)} \quad \max_{x,y} \quad & f_1(x,y) = c_1^T x + c_2^T y && (3.8)\\
\text{s.a} \quad & B_1 x + B_2 y \leq b && (3.9)\\
& x \geq 0,\ y \text{ solução de} && (3.10)\\
& \max_{y} \ f_2(x,y) = d^T y && (3.11)\\
& \text{s.a} \ A_1 x + A_2 y \leq a && (3.12)\\
& \quad y \geq 0 && (3.13)
\end{aligned}
$$

Este algoritmo baseia-se em um método *branch-and-bound* que enumera implicitamente as soluções viáveis de PLDN. A princípio este algoritmo trabalha de uma forma semelhante ao método *branch-and-bound* de Bard e Moore [6] apresentado na seção 2.2.2 (pg.26). No entanto, o procedimento de Hansen *et al.* apresenta algumas diferenças daquele.

O algoritmo de Hansen *et al.* faz uso de uma formulação diferente daquele de Bard e Moore e utiliza uma regra de ramificação baseada nas restrições do problema do seguidor que devem ser ativas em qualquer solução viável de PLDN. Visto que os problemas primal PPS(x) e dual PDS(x) do seguidor serão utilizados na definição do algoritmo, abaixo são dadas suas formulações onde w e v são suas respectivas variáveis de folga associadas.

$$
\begin{aligned}
\text{PPS(x)} \quad \max \quad & d^T y \\
\text{s.a} \quad & A_2 y + w = a - A_1 x \\
& y \geq 0, w \geq 0
\end{aligned}
\qquad\qquad
\begin{aligned}
\text{PDS(x)} \quad \min \quad & (a - A_1 x)^T u \\
\text{s.a} \quad & A_2^T u - v = d \\
& u \geq 0, v \geq 0
\end{aligned}
$$

Defina-se então os vetores $\nu \in \Re^{m_2+n_2}$ e $\omega \in \Re^{m_2+n_2}$, como $\nu = (u, v)$ e $\omega = (w, y)$. Como citado anteriormente, uma solução (x, y, w, u, v) que satisfaça as restrições de PPS(x) e PDS(x), será viável para PLDN quando o vetor (x, y)

pertencer ao conjunto $\{(x,y) \in \Re^{n_1+n_2} : B_1x + B_2y \leq b\}$ e $v^Ty = u^Tw = 0$ ou de uma outra forma $\nu^T\omega = 0$. Hansen *et al.* utiliza uma idéia semelhante a de Bard e Moore [6], que fixa a zero variáveis ν_i e ω_i, no intuito de se chegar a soluções viáveis de PLDN através de um processo enumerativo.

A principal diferença do algoritmo de Hansen *et al.* para o de Bard e Moore está na regra de ramificação. Recorde-se que Bard e Moore escolhia para fixar a zero, em um determinado nó da árvore de enumeração, a variável ν_i ou ω_i que fornecesse o maior valor $\nu_i\omega_i$. Hansen *et al.*, no entanto, utilizam uma nova regra para escolher qual destas variáveis devem ser fixas a zero. A seguir esta nova regra é apresentada em detalhes.

Os autores definem inicialmente um vetor $\alpha \in \Re^{m_2+n_2}$ de variáveis booleanas e associa cada variável α_i a uma variável ω_i do vetor ω. Estas variáveis tem o papel de representarem as restrições do problema do seguidor (incluindo as de não negatividade), indicando se as mesmas estão ativas ou inativas em um dado ponto extremo do conjunto viável relaxado W.

Quando uma variável α_i for igual a 1, sua respectiva variável ω_i estará nula, ou seja, a restrição referente a esta variável ω_i será ativa. De forma contrária, quando uma variável α_i for nula, sua respectiva variável ω_i será estritamente positiva, ou seja, a restrição referente a esta variável ω_i será inativa. Fazendo-se uso destas variáveis α_i, Hansen *et al.* enunciam o seguinte teorema [29, Teorema 4.1]:

Teorema 3.2.1 *Em qualquer solução viável para PLDN as variáveis α_i referentes as variáveis ω_i do problema do seguidor, satisfazem para todo $j \in \{1, 2, \ldots, n_2\}$:*

$$\sum_{i|A_{2ij}>0} \alpha_i \geq 1, \qquad se \;\; d_j > 0 \tag{3.14}$$

$$\alpha_{m_2+j} + \sum_{i|A_{2ij}<0} \alpha_i \geq 1, \qquad se \;\; d_j < 0 \tag{3.15}$$

Nas expressões (3.14) e (3.15), A_{2ij} representa o j-ésimo coeficiente da i-ésima restrição do problema primal do seguidor PPS(x) e d_j representa o j-ésimo elemento do vetor d da função objetivo do problema do seguidor. Defina-se também $\mathcal{R}$ como sendo o conjunto de todas as relações do tipo (3.14) e (3.15).

Perceba que cada relação do tipo (3.14) e (3.15) refere-se aos valores de suas respectivas variáveis ω_i associadas em uma dada solução viável para PLDN. Por

exemplo, tome-se a seguinte relação $(r_k) : \alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3} + \ldots + \alpha_{i_k} + \ldots + \alpha_{i_n} \geq 1$ com $n \leq m_2 + n_2$, onde α_{i_k} é o k-ésimo elemento da relação. Note que esta relação garante que em qualquer solução viável para PLDN, ao menos um elemento desta relação, digamos α_{i_k}, terá valor 1, ou seja, sua respectiva variável ω_i será nula. Isto significa que a restrição referente a esta variável ω_i deverá satisfazer a igualdade em alguma solução viável para (PLDN).

Defina-se então o conjunto ${I_1}^0 = \{1, 2, \ldots, m_2\}$ e seus subconjuntos ${I_1}^+ = \{i \in {I_1}^0 : w_i \geq 0\}$, onde as variáveis w_i podem assumir valores não negativos e ${I_1}^- = \{i \in {I_1}^0 : w_i = 0\}$, onde as variáveis w_i estão fixas a zero, tal que ${I_1}^0 = {I_1}^+ \cup {I_1}^-$. De forma análoga considere o conjunto ${I_2}^0 = \{1, 2, \ldots, n_2\}$ e seus subconjuntos ${I_2}^+ = \{i \in {I_2}^0 : y_i \geq 0\}$, onde as variáveis y_i podem assumir valores não negativos e ${I_2}^- = \{i \in {I_2}^0 : y_i = 0\}$, onde as variáveis y_i estão fixas a zero, tal que ${I_2}^0 = {I_2}^+ \cup {I_2}^-$.

Utilizando-se estes conjuntos é possível definir a seguir o problema (PLDN'), que é correspondente a (PLDN) com algumas variáveis do vetor $\omega = (w, y)$ fixas a zero.

$$
\begin{aligned}
(\text{PLDN}') \quad \max_{x,y} \quad & f_1(x,y) = c_1^T x + c_2^T y && (3.16)\\
\text{s.a} \quad & B_1 x + B_2 y \leq b && (3.17)\\
& x \geq 0,\ y \text{ solução de} && (3.18)\\
& \max_{y} \ f_2(x,y) = d^T y && (3.19)\\
& \text{s.a} \ A_{1i} x + A_{2i} y \leq a_i,\ \forall i \in {I_1}^+ && (3.20)\\
& \quad A_{1i} x + A_{2i} y = a_i,\ \forall i \in {I_1}^- && (3.21)\\
& \quad y_i \geq 0,\ \forall i \in {I_2}^+ && (3.22)\\
& \quad y_i = 0,\ \forall i \in {I_2}^- && (3.23)
\end{aligned}
$$

No problema (PLDN'), A_{1i} e A_{2i} são as i-ésimas linhas das matrizes A_1 e A_2. Note que quando não existe nenhuma variável ω_i fixa a zero, tem-se ${I_1}^+ = {I_1}^0$, ${I_1}^- = \emptyset$, ${I_2}^+ = {I_2}^0$ e ${I_2}^- = \emptyset$, o que torna (PLDN') idêntico a (PLDN). Hansen *et al.*, no entanto, não trabalham com o problema (PLDN') formulado desta maneira. Na verdade os autores utilizam uma outra formulação para (PLDN') onde algumas variáveis y_i são eliminadas. Esta formulação é apresentada a seguir.

Como mencionado anteriormente, quando uma variável α_i é fixa a 1 sua respectiva variável associada ω_i é fixada a zero. Suponha-se então que $i \leq m_2$, sendo assim a variável de folga da i-ésima restrição do seguidor w_i será fixa em zero, o que torna

possível expressar alguma variável y_k ($k \leq n_2$) desta restrição em função das outras variáveis, tal como:

$$y_k = \frac{1}{A_{2ik}} \left(a_i - \sum_{j=1}^{n_1} A_{1ij} x_j - \sum_{\substack{j=1 \\ j \neq k}}^{n_2} A_{2ij} y_j \right) \tag{3.24}$$

onde A_{1ij} e A_{2ij} são os j-ésimos elementos das i-ésimas linhas das matrizes A_1 e A_2. De forma análoga, suponha-se que $m_2 < i \leq m_2 + n_2$. Sendo assim a variável y_{i-m_2} será nula.

Em qualquer dos dois casos, o problema (PLDN′) pode ser simplificado através da eliminação de alguma variável y_k substituindo-se seu valor por 0 ou pelo lado direito da equação (3.24), conforme seja o valor de i.

Defina-se então I_e como sendo o conjunto dos índices i das variáveis y_i eliminadas de (PLDN′) através das equações (3.24). Este problema pode então ser reformulado como:

$$\begin{aligned}
(\text{PLDN}') \quad \max \quad & \tilde{c}_1^T x + \tilde{c}_2^T \tilde{y} && (3.25)\\
\text{s.a} \quad & \tilde{B}_1 x + \tilde{B}_2 \tilde{y} \leq \tilde{b} && (3.26)\\
& x \geq 0,\ \tilde{y} \text{ solução de} && (3.27)\\
& \max\ \tilde{d}^T \tilde{y} && (3.28)\\
& \text{s.a}\ \ \tilde{A}_1 x + \tilde{A}_2 \tilde{y} \leq \tilde{a} && (3.29)\\
& \quad \tilde{y} \geq 0 && (3.30)
\end{aligned}$$

onde o vetor $\tilde{y}$ representa parte do vetor original y onde algumas variáveis y_i foram eliminadas, ou seja, $\tilde{y} = [y_{i1}, y_{i2}, y_{i3}, \ldots, y_{ik}, \ldots, y_{ip}]$, onde $p = n_2 - |I_e| + |I_2^-|$, y_{ik} é o k-ésimo elemento de $\tilde{y}$ e $|I_e|$ e $|I_2^-|$ são os números de elementos existentes nos conjuntos I_e e I_2^- respectivamente.

Os coeficientes das equações (3.25),(3.26),(3.28) e (3.29) marcados com til(˜) são obtidos ao se eliminar de (PLDN′) alguma variável y_i como indicado anteriormente. Como pode haver mais de uma variável y_i eliminada do problema (PLDN′), ou seja $|I_e| + |I_2^-| > 1$, vai-se então definir (E) como sendo o conjunto de equações do tipo (3.24) que expressam o valor de cada variável eliminada y_i em função das restantes.

Note que esta nova formulação de (PLDN′) é mais simples de ser resolvida que a anterior, pois o número de variáveis é menor. Apesar disto, o número de restrições não varia pois perceba que ao se eliminar alguma variável y_i substituindo-a pelo

lado direito da equação (3.24) por exemplo, a restrição de positividade $y_i \geq 0$ faz com que o número de restrições permaneça inalterado.

Uma observação feita pelos autores diz respeito à escolha da variável y_i a ser eliminada de uma determinada restrição. Hansen *et al.* escolhem aquela variável y_i que produza o menor *fill-in*, ou seja, a variável y_i que ao ser substituída pelo lado direito da equação (3.24) gere o menor aumento do número de coeficientes não nulos.

Tendo-se definido o problema (PLDN'), é possível então definir o problema relaxado de (PLDN') denominado (RPLDN'), que consiste de (3.25) a (3.30) omitindo-se (3.28), e o problema do seguidor com algumas variáveis y_i eliminadas, definido por $\text{PPS(x)}' = \arg\max\{\tilde{d}^T\tilde{y} : \tilde{A}_2\tilde{y} \leq \tilde{a} - \tilde{A}_1 x, \tilde{y} \geq 0\}$.

Definidos estes problemas, considere então um certo problema (PLDN') e seus respectivos problemas (RPLDN') e PPS(x)'. Sejam $(\bar{x}, \bar{y}) \in \Re^{n_1} \times \Re^p$ a solução ótima da relaxação do subproblema (RPLDN') e $\hat{y} \in \Re^p$ e $\acute{y} \in \Re^{n_2}$ as soluções ótimas dos problemas PPS$(\bar{x})'$ e PPS$(\bar{x})$ respectivamente. As seguintes observações podem ser feitas:

1. Se o ponto $(\bar{x}, \hat{y})$ for viável para o subproblema (PLDN'), ou seja, satisfizer a restrição (3.26), então $\tilde{c}_1^T\bar{x} + \tilde{c}_2^T\hat{y} \leq \tilde{c}_1^T\bar{x} + \tilde{c}_2^T\bar{y}$. Supondo-se que exista um ponto $(\dot{x}, \dot{y})$ viável para o problema inicial (PLDN) tal que $\tilde{c}_1^T\bar{x} + \tilde{c}_2^T\bar{y} \leq f_1(\dot{x}, \dot{y})$, então não existirá nenhum ponto $(\breve{x}, \breve{y})$ viável para este subproblema (PLDN'), tal que $\tilde{c}_1^T\breve{x} + \tilde{c}_2^T\breve{y} > f_1(\dot{x}, \dot{y})$;

2. O ponto $(\bar{x}, \bar{y})$ será viável para (PLDN') quando $\tilde{d}^T\bar{y} = \tilde{d}^T\hat{y}$. Note, entretanto, que mesmo se $(\bar{x}, \bar{y})$ for viável para (PLDN') não implica dizer que o mesmo será viável para (PLDN), pois (PLDN') possui algumas variáveis do seguidor ω_i fixas a zero e (PLDN) não;

3. Seja então o vetor $\dot{y} \in \Re^{n_2}$ com uma parcela de suas variáveis $\dot{y}_i$ satisfazendo $\dot{y}_i = \bar{y}_{i_k}$, $\forall k \in \{1, 2, \ldots, p\}$ e $i \in \left\{\{1, 2, \ldots, n_2\} \backslash {I_2}^- \cup I_e\right\}$, uma outra parcela satisfazendo $\dot{y}_i = 0, \forall i \in {I_2}^-$ e a última obtida das variáveis $y_i, \forall i \in I_e$, eliminadas de (PLDN') e recuperadas ao se substituir o ponto $(\bar{x}, \bar{y})$ nas equações do conjunto (E). Se o ponto $\dot{y}$ satisfizer $d^T\dot{y} = d^T\acute{y}$ então o ponto $(\bar{x}, \dot{y})$ será viável para PLDN;

4. Se o problema (PLDN') em estudo não mais contiver variáveis $\tilde{y}_i$, então o mesmo passa a ser um problema de programação linear usual.

As conseqüências decorrentes da fixação de alguma variável α_i a 1 foram mostradas em detalhes nas explicações anteriores. Agora será explicado o caso em que alguma variável α_i é fixada a 0.

Como mencionado anteriormente quando uma variável α_i é fixa em 1, sua correspondente variável ω_i é fixa a zero, o que gera o problema (PLDN') citado na pg.46. Entretanto, quando uma variável α_i é fixa a 0, sua correspondente variável ω_i torna-se estritamente positiva ($\omega_i > 0$). Isto significa que existiriam em (PLDN), restrições do tipo $A_{1i}x + A_{2i}y < a_i$ (se $i \leq m_2$) ou $y_{i-m_2} > 0$ (se $m_2 < i \leq m_2 + n_2$). Este fato torna impossível a resolução do problema do seguidor para um certo $\bar{x} \in \Re^{n_1}$ por um método de programação linear convencional, pois o mesmo percorre vértices de seu conjunto viável, o que no caso não poderia ser feito.

Pela teoria da programação linear, é conhecido que para um certo $\bar{x} \in \Re^{n_1}$ a solução ótima $(\bar{y}, \bar{w})$ de PPS($\bar{x}$) e $(\bar{u}, \bar{v})$ de PDS($\bar{x}$), sobre hipótese de compacidade, satisfazem $d^T\bar{y} = (a - A_1\bar{x})^T\bar{u}$, ou seja, $\bar{\omega}^T\bar{\nu} = \bar{w}^T\bar{u} + \bar{y}^T\bar{v} = 0$.

Visto que no ótimo de PPS(x) e PDS(x), para x fixo em um certo $\bar{x}$, a complementaridade deve ser satisfeita, ou seja, $\bar{\omega}^T\bar{\nu} = 0$, Hansen *et al.*, ao invés de trabalharem com uma variável $\omega_i > 0$, trabalham com o problema PDS(x) onde a variável ν_i, correspondente a ω_i, é fixa a 0.

Defina-se então o conjunto ${I_{d_1}}^0 = \{1, 2, \ldots, m_2\}$ e seus subconjuntos ${I_{d_1}}^+ = \{i \in {I_{d_1}}^0 : u_i \geq 0\}$, onde as variáveis u_i podem assumir valores não negativos e ${I_{d_1}}^- = \{i \in {I_{d_1}}^0 : u_i = 0\}$, onde as variáveis u_i estão fixas a zero, tal que ${I_{d_1}}^0 = {I_{d_1}}^+ \cup {I_{d_1}}^-$. De forma análoga considere o conjunto ${I_{d_2}}^0 = \{1, 2, \ldots, n_2\}$ e seus subconjuntos ${I_{d_2}}^+ = \{i \in {I_{d_2}}^0 : v_i \geq 0\}$, onde as variáveis v_i podem assumir valores não negativos e ${I_{d_2}}^- = \{i \in {I_{d_2}}^0 : v_i = 0\}$, onde as variáveis v_i estão fixas a zero, tal que ${I_{d_2}}^0 = {I_{d_2}}^+ \cup {I_{d_2}}^-$.

Utilizando-se estes conjuntos é possível definir a seguir o problema PDS(x)', que é correspondente a PDS(x) com algumas variáveis do vetor $\nu = (u, v)$ fixas a zero, onde u_i e d_i são os i-ésimos elementos dos vetores u e d respecivamente, e ${A_{2i}}^T$ é a i-ésima linha da matriz ${A_2}^T$.

$$\begin{aligned}
\text{PDS(x)}' \quad \min \quad & (a - A_1 x)^T u && (3.31)\\
\text{s.a} \quad & u_i \geq 0,\ \forall i \in {I_{d_1}}^+ && (3.32)\\
& u_i = 0,\ \forall i \in {I_{d_1}}^- && (3.33)\\
& {A_{2i}}^T u \geq d_i,\ \forall i \in {I_{d_2}}^+ && (3.34)\\
& {A_{2i}}^T u = d_i,\ \forall i \in {I_{d_2}}^- && (3.35)
\end{aligned}$$

Perceba que quando muitas variáveis ν_i são fixas a zero, o problema PDS(x)' pode ser inviável, sendo assim o problema (PLDN') também o será, devido a hipótese de compacidade assumida. Esta característica será utilizada em um dos testes do algoritmo *branch-and-bound* de Hansen *et al.*.

A fixação de variáveis α_i a 0 ou 1 acarreta modificações não apenas nos problemas (PLDN) e PDS(x), mas também modifica o conjunto $\mathcal{R}$ de relações to tipo (3.14) e (3.15). Defina-se I_{r_k} como sendo o conjunto dos índices i das variáveis α_i pertencentes a uma relação r_k qualquer, onde r_k é dado por:

$$(r_k) : \alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3} + \ldots + \alpha_{i_k} + \ldots + \alpha_{i_n} \geq 1 \quad \equiv \quad \sum_{i \in I_{r_k}} \alpha_i \geq 1$$

Quando uma variável α_i é fixa a 1, então toda relação (r_k) que contiver esta variável será trivialmente satisfeita. Este fato permite que estas equações sejam eliminadas do conjunto de relações $\mathcal{R}$, sendo assim este conjunto será atualizado para $\mathcal{R} = \mathcal{R} \backslash \{r_k : i \in I_{r_k}\}$.

Quando uma variável α_i é fixa a 0, então a mesma deve ser excluída de toda relação (r_k) que a contiver, ou seja, $I_{r_k} = I_{r_k} \backslash \{i\}$, $\forall r_k \in \mathcal{R}$.

Perceba que, ao se fixar variáveis α_i em 0 ou 1, o conjunto de relações $\mathcal{R}$ é, na verdade, simplificado como explicado anteriormente. Hansen *et al.* utilizam ainda uma forma adicional para simplicar o conjunto $\mathcal{R}$ que consiste na eliminação de relações $(r_k) \in \mathcal{R}$ que sejam redundantes.

Uma relação $r_k \in \mathcal{R}$ é dita redundante quando existir uma relação $r_l \in \mathcal{R}$, tal que o seu conjunto I_{r_l} satisfaça $I_{r_l} \subseteq I_{r_k}$. Isto significa que a relação r_k já é trivialmente satisfeita quando a relação r_l for verificada, o que permite que o conjunto $\mathcal{R}$ sejá atualizado para $\mathcal{R} = \mathcal{R} \backslash \{r_k\}$.

Após estas explicações, o método *branch-and-bound* de Hansen *et al.* pode ser enunciado. No entanto, antes de serem enumerados os passos do algoritmo, é dada uma explicação detalhada sobre a regra de ramificação utilizada pelos autores.

Para um melhor entendimento, recorde-se da regra de ramificação utilizada no algoritmo *branch-and-bound* de Bard e Moore [6] apresentado na seção 2.2.2 (pg.26). Naquele algoritmo os autores resolviam em cada nó da árvore de enumeração o problema (P_1') (pg.25) sem as restrições de complementaridade e fixando-se algumas variáveis ω_i e ν_i a 0. Fazendo-se uso dos conjuntos ${I_1}^+, {I_1}^-, {I_2}^+, {I_2}^-, {I_{d_1}}^+, {I_{d_1}}^-, {I_{d_2}}^+$ e ${I_{d_2}}^-$, definidos anteriormente nas páginas 46 e 50, o problema resolvido nos nós da árvore de enumeração do algoritmo de Bard e Moore, pode ser escrito como:

$$
\begin{aligned}
\max \quad & f_1(x,y) = c_1^T x + c_2^T y \\
\text{s.a} \quad & B_1 x + B_2 y \leq b,\ x \geq 0 \\
& A_{1i} x + A_{2i} y \leq a_i,\ \forall i \in {I_1}^+ \\
& A_{1i} x + A_{2i} y = a_i,\ \forall i \in {I_1}^- \\
& u_i \geq 0,\ \forall i \in {I_{d_1}}^+ \\
& u_i = 0,\ \forall i \in {I_{d_1}}^- \\
& {A_2}^T{}_i u \geq d_i,\ \forall i \in {I_{d_2}}^+ \\
& {A_2}^T{}_i u = d_i,\ \forall i \in {I_{d_2}}^- \\
& y_i \geq 0,\ \forall i \in {I_2}^+ \\
& y_i = 0,\ \forall i \in {I_2}^-
\end{aligned}
$$

Perceba que ao se resolver este problema os autores obtinham o valor de $\omega^T \nu = w^T u + v^T y = u^T(a - A_1 x) - d^T y$. Caso $\omega^T \nu = 0$ então o ponto (x, y) encontrado era viável para PLDN e o problema não necessitaria ser particionado, caso contrário tomava-se ω_i e ν_i tal que $\omega_i \nu_i$ é máximo e obtinha-se dois novos problemas idênticos ao anterior, acrescido das restrições $\omega_i =$ e $\nu_i = 0$ respectivamente.

Hansen *et al.* utilizam uma regra de ramificação mista baseada em duas estratégias. A primeira delas utiliza a mesma idéia de Bard e Moore e a segunda, desenvolvida pelos autores, será explicada adiante.

Como pode ser visto anteriormente, Hansen *et al.* não trabalham com um único problema contendo os conjuntos primal e dual do seguidor, como faz Bard e Moore. Ao contrário, Hansen *et al.* trabalham com os problemas (RPLDN') e PDS(x)' apresentados anteriormente. Esta é uma das principais diferenças entre os dois algoritmos.

Hansen *et al.* resolvem em cada nó da sua árvore de enumeração o problema (RPLDN′) obtendo uma solução $(\bar{x}, \bar{y})$ e determina o ponto $(\bar{x}, \dot{y}) \in \Re^{n_1} \times \Re^{n_2}$, como explicado anteriormente no item 3 (pg.48). Determina-se também o ponto $\bar{u} \in \Re^{m_2}$ viável para o problema PDS$(\bar{x})'$. Tome $\bar{w} = a - A_1\bar{x} - A_2\dot{y}$ e $\bar{v} = {A_2}^T\bar{u} - d$, caso $\bar{\omega}^T\bar{\nu} = \bar{w}^T\bar{u} + \bar{v}^T\dot{y} = 0$, o subproblema (PLDN′) não mais necessita ser particionado pois o ponto $(\bar{x}, \dot{y})$ é viável para PLDN. Caso contrário, Hansen *et al.* determinam, através de um critério específico, uma variável $\bar{\omega}_i$ e $\bar{\nu}_i$ tal que $\bar{\omega}_i\bar{\nu}_i > 0$ e obtêm dois outros subproblemas.

O primeiro deles possui o subproblema (PLDN′) idêntico ao do subproblema anterior acrescido da restrição $\bar{\omega}_i = 0$. Este problema por sua vez é simplificado eliminando-se uma variável y_i, e gerando-se então um novo subproblema (PLDN′). Note que o problema PDS(x)′ permanece inalterado, pois não foi acrescentado ao mesmo nenhuma restrição $\nu_i = 0$.

O segundo subproblema, ao contrário do anterior, deixa o subproblema anterior (PLDN′) inalterado e acrescenta a restrição $\bar{\nu}_i = 0$ ao problema PDS(x)′.

A segunda estratégia de ramificação desenvolvida por Hansen *et al.* baseia-se nas relações do tipo r_k do conjunto $\mathcal{R}$ de relações. Os autores escolhem uma relação $r_k \in \mathcal{R}$ segundo um critério específico, que será mostrado adiante, e realizam ramificações múltiplas ao invés de binária, como na estratégia anterior.

Suponha-se que a relação $(r_k) : \alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3} + \ldots + \alpha_{i_k} + \ldots + \alpha_{i_n} \geq 1$ tenha sido selecionada. Pela estratégia de ramificação de Hansen *et al.* faz-se $(\alpha_{i_1} = 1)$ ou $(\alpha_{i_1} = 0$ e $\alpha_{i_2} = 1)$ ou $(\alpha_{i_1} = \alpha_{i_2} = 0$ e $\alpha_{i_3} = 1)$ ou $(\alpha_{i_1} = \alpha_{i_2} = \alpha_{i_3} = 0$ e $\alpha_{i_4} = 1)$ e assim por diante até ter-se $(\alpha_{i_k} = 0,\ \forall k \in \{1, 2, \ldots, n-1\}$ e $\alpha_{i_n} = 1)$.

Desta forma, são gerados n diferentes subproblemas. Por exemplo, tome os problemas gerados pela ramificação $(\alpha_{i_k} = 0,\ \forall k \in \{1, 2, \ldots, p-1\}$ e $\alpha_{i_p} = 1)$. Estes problemas consistirão de um novo subproblema (PLDN′) idêntico ao do seu pai e acrescido de uma restrição $\omega_{i_p} = 0$ referente a variável α_{i_p} e de um novo problema PDS(x)′, idêntico ao do nó pai, acrescido das restrições $(\nu_{i_k} = 0,\ \forall k \in \{1, 2, \ldots, p-1\})$.

Como se sabe, em um método *branch-and-bound*, aplicado a um problema de maximização quanto mais rápido for determinada uma solução viável para o problema, mais rápido serão obtidos bons limites inferiores e, consequentemente, menos

nós serão explorados. Hansen *et al.* utilizam então uma técnica similar àquela utilizada em programação linear inteira mista (PLIM) onde se determina, dentre todas as variáveis não inteiras do vetor x, por exemplo ($x_i = q$) onde q é um número não inteiro aquela mais promissora a produzir os melhores limites inferiores nos dois subproblemas gerados ao se acrescentar ao problema atual a restrição $x_i \leq \lfloor q \rfloor$ ou $x_i \geq \lceil q \rceil$.

Esta técnica utilizada em (PLIM) quando incorporada ao algoritmo *branch-and-bound* de Hansen *et al.* influenciará na escolha da variável $\bar{\omega}_i$ e $\bar{\nu}_i$, tal que $\bar{\omega}_i \bar{\nu}_i > 0$, dentre aquelas pertencentes aos vetores $\bar{\omega}$ e $\bar{\nu}$ que deverão ser fixas a zero nos problemas (PLDN') e PDS(x)', ou na escolha da ordem de ramificação assumida quando se faz em uma relação $r_k \in \mathcal{R}$.

Perceba que ao se acrescentar uma restrição do tipo $\omega_i = 0$, ou equivalentemente, $\omega_i \leq 0$ ao problema (RPLDN'), o valor ótimo do novo problema gerado é menor que o do problema anterior que era, na verdade, uma relaxação deste. Fazendo-se uso de penalidades utilizadas em programação linear inteira mista [50, pg.154] é possivel se determinar a redução mínima no valor da função objetivo de (RPLDN') ao se acrescentar a restrição $\omega_i \leq 0$.

Considere então a matriz A de restrições do problema (RPLDN') atual e particione a mesma em $A = [B\ N]$, onde B é uma matriz inversível e N uma matriz de dimensões apropriadas. Considere ainda os conjuntos (V_B) e (V_{NB}) como sendo os conjuntos dos índices das variáveis básicas e não básicas associados ao problema. Seja $(B^{-1}g)_i$ o i-ésimo elemento do vetor $(B^{-1}g)$, onde g é o vetor RHS (*right hand side*) do problema (RPLDN'), e $(B^{-1}N)_{ij}$ o j-ésimo elemento da i-ésima linha da matriz $(B^{-1}N)$. Considere ainda $(c_B{}^T B^{-1}N - c_N{}^T)_j$ como sendo o j-ésimo elemento do vetor de custos reduzidos $(c_B{}^T B^{-1}N - c_N{}^T)$, onde $c^T = [c_B{}^T\ c_N{}^T]$ é o vetor de coeficientes da função objetivo do líder particionado de acordo com os índices das variáveis básicas e não básicas.

Considerando-se que B é uma base ótima para o problema (RPLDN') então o decréssimo mínimo p_i no valor da função objetivo, ao se acrescentar uma restrição $\omega_i \leq 0$ ao quadro ótimo de (RPLDN'), pode ser calculado através da primeira

iteração do método dual simplex como sendo:

$$p_i = (B^{-1}g)_i \min_{j \in (V_{NB}):(B^{-1}N)_{ij}>0} \left\{ \frac{(c_B{}^T B^{-1}N - c_N{}^T)_j}{(B^{-1}N)_{ij}} \right\}$$

Perceba que o cálculo de p_i é extremamente simples, visto que estes dados fazem parte do quadro ótimo do problema (RPLDN'). Hansen *et al.* utilizam estas penalidades para melhorar as regras de ramificação e poda de seu algoritmo.

Hansen *et al.* consideraram em seus testes computacionais várias regras de ramificação, dentre elas a regra mista apresentada anteriormente. Seja então $\bar{\omega} \in \Re^{m_2+n_2}$ a solução ótima do problema (RPLDN') nas dimensões originais e $\bar{\nu} \in \Re^{m_2+n_2}$ um ponto extremo do conjunto viável de PDS(x)'. Sendo assim os autores escolhem para fixar em zero aquela variável $\bar{\omega_i}$ que possua a maior penalidade p_i dentre aquelas que satisfaçam $\bar{\omega_i}\bar{\nu_i} > 0$.

No processo de poda (*prunning*) estas penalidades também são úteis, como apresentado a seguir. Defina-se o parâmetro Π como:

$$\Pi = \max_{\forall r_k \subseteq \mathcal{R}} \{\min\{p_i : i \in I_{r_k}\}\}$$

Considere então F^* como sendo o valor ótimo do problema (RPLDN'). Perceba que o parâmetro Π indica qual será o menor decréscimo no valor de F^* ao se satisfazerem todas as relações $r_k \subseteq \mathcal{R}$.

Este parâmetro exerce um importante papel no processo de poda. De fato, suponha que o melhor limite inferior conhecido até o momento seja f_{opt}. Sendo assim, se $F^* - \Pi \leq f_{opt}$, então o problema (PLDN') não mais precisa ser particionado, visto que não será possível se encontrar através deste uma solução viável para (PLDN) que seja maior que f_{opt}.

Finalmente, após estas explicações é possível apresentar os passos do algoritmo:

Algoritmo (Hansen *et al.* [29])

Passo 1 (Inicialização) Obtenha uma solução inicial (x_h, y_h) viável para PLDN com uma heurística. Inicialize a melhor solução encontrada $(x^\star, y^\star)$ com (x_h, y_h) e faça $f_{opt} = f_1(x^\star, y^\star)$. Se nehuma solução heurística pode ser encontrada, inicialize $(x^\star, y^\star)$ com um ponto arbitrário e faça $f_{opt} = -\infty$. Considere inicialmente todas as variáveis α_i, $i \in \{1, 2, \ldots, m_2+n_2\}$, livres,

ou seja, nenhuma variável está fixa a 0 ou 1. Faça $\mathcal{R} = \emptyset$ e considere o subproblema atual (PLDN') igual a PLDN.

Passo 2 Resolva o problema (RPLDN'), obtendo uma solução $(\bar{x}, \bar{y})$. Se $f_{opt} \geq \tilde{c}_1^T \bar{x} + \tilde{c}_2^T \bar{y}$, vá para o passo 13 (*backtracking*).

Passo 3 Determine uma solução viável $\bar{u}$ para o problema PDS(x)'. Se não existe solução viável, vá para o passo 13.

Passo 4 Verifique se $(\bar{x}, \bar{y})$ é viável para o problema atual (PLDN'): resolva PPS$(\bar{x})'$, obtendo $\hat{y}$; se $\tilde{d}^T \bar{y} = \tilde{d}^T \hat{y}$, então $(\bar{x}, \bar{y})$ é viável para (PLDN'); senão vá para o passo 6.

Passo 5 Determine, a partir de $(\bar{x}, \bar{y})$, o ponto $(\bar{x}, \dot{y}) \in \Re^{n_1} \times \Re^{n_2}$, como indicado no item 3 (pg.48). Verifique se $(\bar{x}, \dot{y})$ é viável para PLDN: resolva PPS$(\bar{x})$, obtendo $\acute{y}$; se $d^T \dot{y} = d^T \acute{y}$ então $(\bar{x}, \dot{y})$ é viável para (PLDN); senão vá para o passo 6. Atualize f_{opt} e $(x^\star, y^\star)$ se $\tilde{c}_1^T \bar{x} + \tilde{c}_2^T \bar{y} > f_{opt}$; vá para o passo 13.

Passo 6 Calcule todas as penalidades p_i associadas as variáveis ω_i estritamente positivas na solução ótima de (RPLDN'). Faça as outras penalidades p_i iguais a zero. Calcule então o valor de Π como indicado na página 54. Se $f_{opt} \geq f_1(\bar{x}, \bar{y}) - \Pi$, vá para o passo 13.

Passo 7 Se (RPLDN') é inviável, vá para o passo 13.

Passo 8 Para cada i tal que $f_{opt} \geq f_1(\bar{x}, \bar{y}) - p_i$, fixe $\alpha_i = 0$ e atualize $\mathcal{R}$.

Passo 9 Se $\mathcal{R}$ contém uma relação r_k tal que $\alpha_i = 0$ para todo $i \in I_{r_k}$, vá para o passo 13.

Passo 10 Calcule todas as relações do tipo (3.14) e (3.15) relacionadas ao problema (PLDN') atual e as inclua no conjunto de relações $\mathcal{R}$, caso as mesmas sejam não redundantes. Retire de $\mathcal{R}$ todas as relações que se tornarem redundantes.

Passo 11 Se $\mathcal{R}$ contém uma relação r_k tal que $\alpha_j = 0$ para todo $j \in I_{r_k} - \{i\}$, faça $\alpha_i = 1$. Elimine uma variável y_j, modificando o subproblema corrente, atualize o conjunto $\mathcal{R}$ e retorne ao passo 2.

Passo 12 (Ramificação) Aplique a regra de ramificação selecionada de modo a escolher ou uma variável α_i livre ou uma relação $r_k \in \mathcal{R}$ onde todas as variáveis

α_i ($i \in I_{r_k}$) estão livres. No primeiro caso, ramifique com $\alpha_i = 1$. No segundo, ramifique fazendo a primeira variável α_i em r_k igual a 1. Elimine uma variável y_j, obtendo um novo subproblema e volte ao passo 2.

Passo 13 (*Backtracking*) Se a ramificação ocorreu em uma variável, determine a última variável $\alpha_i = 1$ escolhida para ramificação; faça $\alpha_i = 0$ e torne livres todas as variáveis α_j fixadas em 0 depois que α_i foi fixada em 1. Caso contrário, considere a última relação r_k para a qual menos de $|I_{r_k}|$ ramos tenham sido explorados; considere o próximo ramo. Se não existe tal variável ou relação, pare. Senão, atualize o subproblema corrente e retorne ao passo 2.

Hansen *et al.* testaram em seus experimentos computacionais várias regras de ramificação, como já mencionado anteriormente. Na implementação realizada neste trabalho foi utilizada a regra de ramificação **BR6** [29, pg.1208], pois a mesma apresentou os melhores resultados computacionais. Esta regra é apresentada a seguir.

BR6 - Ramificação Mista

(i) Selecione uma relação $r_k \in \mathcal{R}$ com duas variáveis α_{i_1} e α_{i_2}, e com suas correspondentes variáveis ω_{i_1} e ω_{i_2} na base ($\omega_{i_1} > 0$ e $\omega_{i_2} > 0$), tal que $\sum_{i \in I_{r_k}} \omega_i \nu_i$ seja máximo;

(ii) Se não existir tal relação, então ramifique na variável α_i com a maior penalidade p_i dentre todas aquelas com $\omega_i \nu_i > 0$.

Os vetores ω e ν utilizados nesta regra de ramificação são provenientes da solução ótima $\bar{\omega}$ do problema (RPLDN') e do ponto viável $\bar{\nu}$ no problema PDS(x)'.

Os autores utilizam no Passo 1 do algoritmo uma heurística para se determinar uma solução inicial viável para o problema. Na ausência de restrições no primeiro nível Hansen *et al.* resolvem o problema (RPLDNP) onde a função objetivo é substituída por $\lambda {c_1}^T x + (1 - \lambda) {c_2}^T y$, onde $\lambda = \frac{n_1 + n_2}{n_1 + 2n_2}$ e n_1 e n_2 são as dimensões dos vetores c_1 e c_2 respectivamente.

Finalmente, Hansen *et al.* mostram que o algoritmo pára em um número finito de iterações e encontra um ótimo global para PLDN [29, Teorema 4.3].

Este procedimento foi comparado com o algoritmo de Bard e Moore [6], como mencionado anteriormente, e superou o mesmo em todos os testes computacionais realizados. Hansen *et al.* enumeram então as possíveis razões para o melhor desempenho do seu algoritmo em relação ao último:

1. As relações do tipo (3.14) e (3.15) em problemas esparsos muitas vezes são compostas de apenas uma variável α_i permitindo então a eliminação de variáveis, sem no entanto necessitar de ramificação;

2. A resolução dos problemas (RPLDN′) e PDS(x)′ separados ao invés de juntos, como fazem Bard e Moore (pg.51), reduzem os tamanhos dos problemas lineares;

3. O uso de uma regra de ramificação mista e das penalidades.

Capítulo 4

Modificações Propostas

Este capítulo traz a verdadeira contribuição deste trabalho. Nele são apresentados os algoritmos de penalidade e *branch-and-bound* modificados. Estes novos algoritmos são modificações do algoritmo de Campêlo [16] e do algoritmo de Hansen *et al.* [29] apresentados no capítulo anterior (pgs. 34 e 44, respectivamente).

As alterações propostas nestes algoritmos foram sugeridas com base nos experimentos computacionais realizados no Capítulo 5, ao se comparar o algoritmo de penalidades de Campêlo com o algoritmo *branch-and-bound* de Hansen *et al.*. Os resultados computacionais obtidos confirmaram a eficiência do algoritmo *branch-and-bound* em detrimento do algoritmo de penalidade.

No algoritmo de penalidades foi observado que a maior parte do esforço computacional realizado se deveu ao método *outer approximation* adaptado utilizado para resolver um subproblema de complementaridade linear. No caso do algoritmo *branch-and-bound*, apesar de sua eficiência, foi observado que, em alguns problemas, o mesmo consumia boa parte do tempo para determinar bons limites inferiores. Baseado nestas observações, as seguintes modificações foram propostas.

4.1 Algoritmo de penalidade modificado

Recorde-se que no algoritmo global de Campêlo (pg.64), o procedimento *outer approximation* adaptado (pg.42) é utilizado para resolver, no Passo 5, o seguinte problema de complementaridade linear (pg.40):

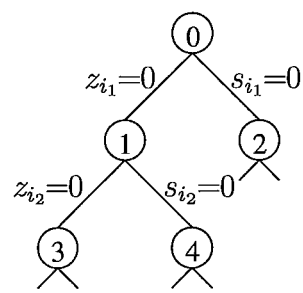
$$\begin{array}{lll} (\text{PCL}') & \min & g(z,s) = s^T z \\ & \text{s.a} & (z,s) \in Z' \times S \end{array}$$

onde $Z' = \{z \in \Re^n : c^T z = F^* + \varepsilon, Az = a, z^T = (x^T, y^T, w^T) \geq 0\}$ e $S = \{s \in \Re^n : Ds = d, s^T = (0, v^T, u^T) \geq 0\}$. No intuito de facilitar o entendimento, a notação utilizada é recordada: $A = [A_1\ A_2\ I_{m_2}] \in \Re^{m_2 \times n}$ e $D = [0\ -I_{n_2}\ A_2^T] \in \Re^{n_2 \times n}$ são matrizes bloco, $c^T = (c_1^T, c_2^T, 0) \in \Re^n$ um vetor, $n = n_1 + n_2 + m_2$, I_k uma matriz identidade de ordem k e 0 uma matriz nula de dimensões apropriadas.

Recorde-se também que, para o algoritmo global de Campêlo, é necessário apenas que o algoritmo *outer approximation* adaptado retorne um ponto $(\hat{z}, \hat{s})$ tal que $\hat{z}^T\hat{s} < \bar{z}^T\bar{s}$, onde $(\bar{z}, \bar{s})$ é um ponto de equilíbrio satisfazendo $\bar{z}^T\bar{s} > 0$, para que o Algoritmo 2 (pg.38) de equilíbrio possa ser reinicializado.

Neste trabalho é proposto um procedimento enumerativo que determina um ponto $(\hat{z}, \hat{s}) \in Z' \times S$, que satisfaça $\hat{z}^T\hat{s} = 0$, ou verifique que o mesmo não existe. Este novo algoritmo substituirá o *outer approximation* adaptado no algoritmo global de Campêlo.

O processo enumerativo proposto pode então ser visualizado através da seguinte árvore binária:



onde z_j e s_j são as j-ésimas variáveis dos vetores z e s respectivamente para $j \in \{1, 2, \ldots, n\}$.

Em cada nó k da árvore de enumeração são associados os subconjuntos $Z'_k \subseteq Z'$ e $S_k \subseteq S$ que definem um subproblema (PCL$'_k$) semelhante ao problema (PCL'), onde algumas variáveis z_j e s_j foram fixas a zero. Além disso, em cada problema (PCL$'_k$), é encontrado um ponto $(z^k, s^k) \in Z'_k \times S_k$, se os conjuntos Z'_k e S_k forem não vazios.

No nó inicial ($k = 0$), define-se $Z'_0 = Z'$, $S_0 = S$ e (PCL$'_0$)=(PCL'). Considere agora um nó $k \geq 1$ que é gerado apartir de seu nó pai p pela fixação de uma variável z_i ou s_i a zero. Se a variável z_i foi fixa a zero então $Z'_k = Z'_p \cap \{z \in \Re^n : z_i = 0\}$ e $S_k = S_p$. Caso seja uma variável s_i fixa a zero então $S_k = S_p \cap \{s \in \Re^n : s_i = 0\}$

e $Z'_k = Z'_p$. Por exemplo, olhando-se para a árvore binária anterior, tem-se $Z'_1 = Z'_0 \cap \{z \in \Re^n : z_{i_1} = 0\}$ ou ainda $S_4 = S_1 \cap \{s \in \Re^n : s_{i_2} = 0\}$.

Além disso, em cada nó k da árvore, é aplicado um procedimento semelhante ao algoritmo *Mountain Climbing* de Konno [40], partindo-se de um ponto z^k ou s^k. Este procedimento é chamado de *Algoritmo de Ponto de Equilíbrio Adaptado* (APEA). Na verdade (APEA) é uma simplificação do Algoritmo 2 de equilíbrio, para encontrar um ponto estacionário do problema (PCL$'_k$).

A seguir o procedimento (APEA), partindo-se de um ponto $\bar{z} := z^k$, é definido. Note que um procedimento similar pode ser estabelecido, partindo-se de um ponto $\bar{s} := s^k$. A hipótese de que $Z'_k \times S_k \neq \emptyset$ é considerada.

Algoritmo (*Ponto de Equilíbrio Adaptado*)

Passo 0 Faça $z^0 = \bar{z}$ e $l = 0$;

Passo 1 Resolva o problema $\min\{(z^l)^T s : s \in S_k\}$ obtendo uma solução s^l;

Passo 2 Resolva o problema $\min\{(s^l)^T z : z \in Z'_k\}$ obtendo uma solução z^{l+1}. Se $(s^l)^T z^l = (s^l)^T z^{l+1}$, PARE e retorne $(\tilde{z}, \tilde{s}) = (z^l, s^l)$;

Passo 3 Resolva o problema $\min\{(z^{l+1})^T s : s \in S_k\}$ obtendo uma solução s^{l+1}. Se $(s^l)^T z^{l+1} = (s^{l+1})^T z^{l+1}$, PARE e retorne $(\tilde{z}, \tilde{s}) = (z^{l+1}, s^l)$. Caso contrário faça $l = l + 1$ e vá para o passo 2.

As seguintes proposições referentes ao algoritmo descrito acima são válidas.

Proposição 4.1.1 *O algoritmo (APEA) pára em um número finito de iterações.*

Prova Perceba que pelos Passos 2 e 3 a função $g(z, s) = s^T z$ nunca aumenta. Sendo assim, tomando-se dois pontos (z^l, s^l) e (z^{l+1}, s^{l+1}) obtidos em iterações consecutivas do algoritmo, tem-se $g(z^l, s^l) \geq g(z^{l+1}, s^{l+1})$. Como o número de vértices do conjunto $Z'_k \times S_k$ é finito, o critério de parada será eventualmente satisfeito. ∎

Proposição 4.1.2 *O ponto $(\tilde{z}, \tilde{s})$ encontrado pelo algoritmo (APEA) satisfaz as condições de Karush-Kuhn-Tucker (KKT) para o problema (PCL$'_k$).*

Prova Sejam A_k e D_k as matrizes que definem os poliedros Z'_k e S_k respectivamente, ou seja, $Z'_k = \{z \in \Re^n : A_k z = \bar{a}, z \geq 0\}$ e $S_k = \{s \in \Re^n : D_k s = \bar{d}, s \geq 0\}$, onde

os vetores $\bar{a}$ e $\bar{d}$ incorporam o lado direito das restrições acrescentadas. Seja então $(\tilde{z}, \tilde{s})$ o ponto retornado por (APEA). O mesmo satisfaz:

$$\min\{\tilde{z}^T s : s \in S_k\} = \tilde{z}^T \tilde{s} = \min\{\tilde{s}^T z : z \in Z'_k\}. \tag{4.1}$$

Pelo lado esquerdo de (4.1) tem-se que $\exists(\tilde{\mu}, \tilde{\lambda})$ tal que as seguintes equações são satisfeitas:

$$\tilde{z} + {D_k}^T \tilde{\mu} - \tilde{\lambda} = 0, \quad \tilde{\lambda}^T \tilde{s} = 0.$$

De forma análoga, pelo lado direito de (4.1) tem-se que $\exists(\tilde{\beta}, \tilde{\gamma})$ tal que as seguintes equações são satisfeitas:

$$\tilde{s} + {A_k}^T \tilde{\beta} - \tilde{\gamma} = 0, \quad \tilde{\gamma}^T \tilde{z} = 0.$$

Estas equações juntamente com $\tilde{z} \in Z'_k$ e $\tilde{s} \in S_k$ fornecem as condições de (KKT) para o problema (PCL$'_k$). ■

Em particular, quando um dos problemas em (4.1) possuir solução única, então o ponto $(\tilde{z}, \tilde{s})$ é um ponto de mínimo local estrito de (PCL$'_k$), como mostra o resultado abaixo:

Proposição 4.1.3 *Seja $(\tilde{z}, \tilde{s})$ o ponto retornado por (APEA). Se $\tilde{z} = \arg\min\{\tilde{s}^T z : z \in Z'_k\}$ ou $\tilde{s} = \arg\min\{\tilde{z}^T s : s \in S_k\}$ então $(\tilde{z}, \tilde{s})$ é um ótimo local estrito de (PCL$'_k$).*

Prova Dado que $Z'_k \times S_k$ é um conjunto convexo, a derivada direcional de $g(z, s) = z^T s$ no ponto $(\tilde{z}, \tilde{s})$ com respeito a qualquer direção viável é dada por:

$$\nabla g(\tilde{z}, \tilde{s})^T \begin{pmatrix} z - \tilde{z} \\ s - \tilde{s} \end{pmatrix} = \tilde{s}^T (z - \tilde{z}) + \tilde{z}^T (s - \tilde{s}),$$

onde $(z, s) \in Z'_k \times S_k$. Por (4.1) tem-se que $\tilde{s}^T z \geq \tilde{s}^T \tilde{z}$ para todo $z \in Z'_k$ e $\tilde{z}^T s \geq \tilde{z}^T \tilde{s}$ para todo $s \in S_k$. Mais ainda, pela hipótese segue-se que $\tilde{s}^T z > \tilde{s}^T \tilde{z}$ para todo $z \in Z'_k$ ou $\tilde{z}^T s > \tilde{z}^T \tilde{s}$ para todo $s \in S_k$. Logo, $\tilde{s}^T (z - \tilde{z}) + \tilde{z}^T (s - \tilde{s}) > 0$ para todo $(z, s) \in Z'_k \times S_k$. Portanto $(\tilde{z}, \tilde{s})$ é um ótimo local de (PCL$'_k$). ■

Desconsiderando-se a hipótese da proposição anterior, pode-se obter uma condição de otimalidade local mais fraca. De fato, mostra-se que $(\tilde{z}, \tilde{s})$ é um ponto de mínimo local *estrela* de (PCL$'_k$), que é uma solução básica viável $(\tilde{z}, \tilde{s})$ tal que $g(\tilde{z}, \tilde{s}) \leq g(z, s)$ para toda solução básica viável adjacente (z, s) (ver definição em Júdice e Faustino [37]). Este resultado é estabelecido na proposição a seguir.

Proposição 4.1.4 *O ponto $(\tilde{z}, \tilde{s})$ encontrado pelo algoritmo (APEA) é um mínimo local estrela para o problema (PCL'_k).*

Prova Seja $(\tilde{z}, \tilde{s})$ o ponto retornado por (APEA), que, portanto, é um vértice de $Z \times S$ satisfazendo (4.1). Seja (z, s) um vértice de $Z \times S$ adjacente a $(\tilde{z}, \tilde{s})$. Então, $z = \tilde{z}$ ou $s = \tilde{s}$. No primeiro caso, segue-se de (4.1) que $z^T s = \tilde{z}^T s \geq \tilde{z}^T \tilde{s}$. No segundo, obtém-se similarmente que $z^T s = z^T \tilde{s} \geq \tilde{z}^T \tilde{s}$. Logo, o resultado segue. ∎

Definam-se os conjuntos I_z e I_s de índices i das variáveis z_i e s_i fixas a 0, respectivamente. Então o algoritmo enumerativo proposto para encontrar um ponto complementar no conjunto $Z' \times S$, ou verificar que o mesmo não existe, é descrito a seguir.

Algoritmo (*Enumerativo*)

Passo 0 Faça $I_z = \emptyset$, $I_s = \emptyset$ e $g_{opt} = \infty$. Faça $k = 0$;

Passo 1 Se $k = 0$ vá para o Passo 2. Caso contrário atualize $Z'_k = Z' \cap \{z \in \Re^n : z_i = 0, \forall i \in I_z\}$ e $S_k = S \cap \{s \in \Re^n : s_i = 0, \forall i \in I_s\}$. Se $Z'_k = \emptyset$ ou $S_k = \emptyset$ vá para o Passo 6.

Passo 2 Obtenha através da primeira fase do método simplex os vértices $\bar{z}$ e $\bar{s}$ dos conjuntos Z'_k e S_k respectivamente;

Passo 3 Aplique (APEA) apartir de um ponto $\bar{z}$ obtendo um ponto $(\tilde{z}, \tilde{s})$. Se $g_{opt} \geq \tilde{z}^T \tilde{s}$ então faça $g_{opt} = \tilde{z}^T \tilde{s}$ e $(\hat{z}, \hat{s}) = (\tilde{z}, \tilde{s})$. Aplique (APEA) apartir de um ponto $\bar{s}$ obtendo um ponto $(\tilde{z}, \tilde{s})$. Se $g_{opt} \geq \tilde{z}^T \tilde{s}$ então faça $g_{opt} = \tilde{z}^T \tilde{s}$ e $(\hat{z}, \hat{s}) = (\tilde{z}, \tilde{s})$;

Passo 4 Se $g_{opt} = 0$, PARE;

Passo 5 Selecione uma variável z_i tal que o produto $\bar{z}_i \bar{s}_i$ seja máximo. Fixe esta variável a 0 e faça $I_z = I_z \cup \{i\}$. Faça $k = k + 1$ e vá para o Passo 1;

Passo 6 Encontre a última variável z_i ramificada acima e fixa a 0. Se não existe tal variável, PARE. Caso contrário libere a variável z_i, atualize $I_z = I_z \backslash \{i\}$, fixe a variável s_i a 0 e atualize $I_s = I_s \cup \{i\}$. Além disto, libere todas as variáveis s_j, com $j \in I_{s^i}$, onde $I_{s^i} \subset I_s$ é o subconjunto de índices j das

variáveis s_j fixas a 0 após z_i ter sido fixa a 0, atualize $I_s = I_s \backslash I_{s^i}$, faça $k = k + 1$ e vá para o Passo 1.

A seguinte proposição referente ao algoritmo enumerativo descrito é válida.

Proposição 4.1.5 *O Algoritmo Enumerativo pára em um número finito de passos e retorna um ponto $(\hat{z}, \hat{s}) \in Z' \times S$, tal que $\hat{z}^T \hat{s} = 0$, ou conclui que o mesmo não existe.*

Prova Perceba que o algoritmo Enumerativo pára em no máximo $2^{m_2+n_2}$ iterações, que é o número máximo de subproblemas do tipo (PCL$'_k$) que podem ser gerados ao se fixar variáveis $z_i \in Z'$ ou $s_i \in S$ a 0. Note também que se existe algum ponto $(z, s) \in Z' \times S$ complementar o algoritmo o encontrará pois os pontos que satisfaçam esta condição estão sendo implicitamente enumerados. ■

Este algoritmo Enumerativo possui certas semelhanças com o algoritmo enumerativo de Júdice e Faustino [37], utilizado para resolver um problema específico de complementaridade linear apresentado anteriormente (pg.30).

O algoritmo enumerativo de Júdice e Faustino [37] possui sua regra de ramificação baseada na fixação de variáveis z_i e s_i a 0, tal como é feito no algoritmo enumerativo proposto neste trabalho. No entanto, algumas das principais diferenças são citadas a seguir.

1. Tamanho dos problemas considerados. No caso de Júdice e Faustino os conjuntos S e $\tilde{Z}$ (ressalta-se que os autores consideram este conjunto ao invés de Z') fazem parte do mesmo problema, ao contrário do método utilizado neste trabalho que considera os problemas separados. Hansen *et al.* [29] também utilizam a estratégia de se trabalhar com os conjuntos separados ao contrário Bard e Moore [6], indicando como sendo esta uma das principais razões do excelente desempenho de seu algoritmo em comparação ao último;

2. Critério de poda (*prunning*). Ao invés de se fixar uma variável z_i ou s_i a 0, e verificar se os conjuntos Z'_k ou S_k são vazios, Júdice e Faustino calculam o mínimo de z_i ou s_i sobre os conjuntos Z'_k ou S_k. Se este mínimo for estritamente positivo então o nó pode ser podado, caso contrário é fixa a 0 uma variável z_i ou s_i conforme for o caso;

3. Utilização do algoritmo (APEA) para acelerar a convergência do algoritmo enumerativo.

Ressalte-se que a idéia do algoritmo (APEA) também é usada no método *branch-and-bound* de Audet *et al.* [5], que é similar ao algoritmo de Hansen *et al.* [29] adaptado a um caso particular de PLDN que é o problema de programação bilinear (pg.8).

Audet *et al.* utilizam um procedimento semelhante ao algoritmo (APEA) no intuito de se determinar melhores limites inferiores em sua árvore de enumeração e assim acelerar a convergência de seu algoritmo.

O método *outer approximation* adaptado, utilizado no algoritmo global proposto por Campêlo [16], é então substituído por este novo algoritmo Enumerativo como descrito a seguir.

Algoritmo (Algoritmo de Penalidade Modificado)

Passo 1 Se $S = \emptyset$, vá para o passo 6. Senão encontre $\hat{s} \in S$ e considere $\tilde{Z} = Z$. Faça $k = 0$.

Passo 2 Se $\tilde{Z} = \emptyset$, vá para o passo 6.

Passo 3 Aplique o Algoritmo 2 (pg.38) de equilíbrio a partir de $\hat{s}$. Se $\tilde{P}(M)$ é ilimitado para todo $M \geq 0$, pare: PLDNP é ilimitado. Do contrário, seja $(\bar{z}, \bar{s})$ o ponto de equilíbrio encontrado.

Passo 4 Se $\bar{s}^T\bar{z} = 0$, então $(\bar{z}, \bar{s})$ é solução viável de PLDNP; faça $(z^\star, s^\star) = (\bar{z}, \bar{s})$ e $F^\star = F(\bar{z}, \bar{s})$; aplique um corte linear, definindo $\tilde{Z} = Z \cap \{z \in \Re^n : c^T z \geq F^\star + \varepsilon\}$, com $\varepsilon > 0$; faça $\hat{s} = \bar{s}$ e $k = k + 1$; volte ao passo 2.

Passo 5 Se $\bar{s}^T\bar{z} > 0$, aplique o *Algoritmo Enumerativo* (pg.62) obtendo um ponto $(\hat{z}, \hat{s})$. Se $\hat{z}^T\hat{s} = 0$ vá para o Passo 3;

Passo 6 Pare. Se $k = 0$, PLDN é inviável. Senão, $(x^\star, y^\star)$ é uma ε-solução global de PLDNP.

Note que a idéia de se construir uma árvore de enumeração a cada vez que se encontre um ponto de equilíbrio inviável para (PLDNP) poderia parecer a princípio

extremamente caro do ponto de vista computacional. No entanto, tão logo se encontre um ponto $(\hat{z}, \hat{s})$, tal que $\hat{z}^T\hat{s} = 0$, o algoritmo enumerativo pode parar sem a necessidade de se explorar os nós restantes da árvore de enumeração, como acontece em um método *branch-and-bound* convencional.

A única excessão ocorre na última iteração do algoritmo de penalidade modificado, na qual é necessário checar a otimalidade do ótimo global mostrando-se que não existe um outro ponto viável para (PLDNP) com um valor maior na função objetivo do líder. Neste caso, a árvore de enumeração necessita ser completamente explorada, parando apenas pelos critérios de poda adotados.

4.1.1 Outras abordagens

Nesta seção é apresentada uma idéia proposta por Amouzegar e Moshirvaziri [2] para se determinar um ponto viável para (PLDNP) satisfazendo um corte semelhante àquele utilizado por Campêlo [16]. Esta estratégia utilizada pelos autores é apresentada neste trabalho no intuito de motivar trabalhos futuros, que possam vir a substituir o algoritmo Enumerativo proposto.

De maneira similar a estratégia de Campêlo, o algoritmo global de Amouzegar e Moshirvaziri [2] ao encontrar uma solução local $(\bar{z}, \bar{s})$ de (PLDNP), resolve o seguinte problema de complementaridade linear:

$$\begin{array}{lll} (\overline{\text{PCL}}) & \min & s^T z \\ & \text{s.a} & (z, s) \in \bar{Z} \times S \end{array}$$

onde $\bar{Z} = \{z \in \Re^n : c^T z = c^T \bar{z} = \xi, Az = a, z^T = (x^T, y^T, w^T) \geq 0\}$ e $S = \{s \in \Re^n : Ds = d, s^T = (0, v^T, u^T) \geq 0\}$.

Ao tentar resolver este problema, os autores procuram um ponto $(\hat{z}, \hat{s}) \neq (\bar{z}, \bar{s})$, tal que $\hat{z}^T\hat{s} = 0$. Se este ponto não existe os autores concluem que $(\bar{z}, \bar{s})$ é um ótimo global de (PLDNP), caso contrário um algoritmo local é inicializado a partir de $(\hat{z}, \hat{s})$ na intenção de se encontrar um outro ótimo local $(\tilde{z}, \tilde{s})$ para (PLDNP), satisfazendo $c^T\tilde{z} > c^T\hat{z}$.

Para resolver $(\overline{\text{PCL}})$ os autores determinam inicialmente uma função linear $\tilde{c}^T z$, tal que $\bar{z} = \arg\max\{\tilde{c}^T z : z \in \bar{Z}\}$. A determinação do vetor $\tilde{c}$ não é explicada em [2] pelos autores, no entanto neste trabalho é apresentada a seguinte proposição, nela são utilizados a matriz $B = \begin{bmatrix} A_1 & A_2 \\ {c_1}^T & {c_2}^T \end{bmatrix}$, o vetor $b^T = [a^T \ \xi]$ e as variáveis

$\eta^T = [w^T\ 0]$ e $\gamma^T = [x^T\ y^T]$.

Proposição 4.1.6 *Dado um ponto $\bar{\gamma} \in G = \{\gamma \in \Re^{n_1+n_2} : B\gamma \leq b, \gamma \geq 0\}$, o vetor $\tilde{c}$ tal que $\bar{\gamma} = \arg\max\{\tilde{c}^T\gamma : \gamma \in G\}$ é dado por qualquer combinação linear positiva das restrições de G, incluindo as restrições de não negatividade, ativas em $\bar{\gamma}$.*

Prova Tem-se abaixo o problema $(\dot{P})$, cuja solução ótima é $\bar{\gamma}^T = [\bar{x}^T\ \bar{y}^T]$, e ao lado o problema equivalente a $(\dot{P})$, obtido ao se substituir o mesmo por suas condições de Karush-Kuhn-Tucker (KKT).

$$
\begin{array}{llll}
(\dot{P}) & \max & \tilde{c}^T\gamma & \\
 & \text{s.a} & B\gamma + \eta = b & \longrightarrow \\
 & & \gamma \geq 0, \eta \geq 0 &
\end{array}
\qquad
\begin{array}{ll}
KKT-(\dot{P}) & B\gamma + \eta = b \\
 & B^T\mu - \nu = \tilde{c} \\
 & \gamma^T\nu = \mu^T\eta = 0 \\
 & \gamma \geq 0, \eta \geq 0, \mu \geq 0, \nu \geq 0
\end{array}
$$

Da formulação de (KKT) do problema $(\dot{P})$, pode ser inferido que o valor de $\tilde{c}$ deve satisfazer:

$$\tilde{c} = \sum_{i:\eta_i=0} B_i\mu_i + \sum_{j:\gamma_j=0} -I_j\nu_j \tag{4.2}$$

onde B_i e μ_i são a i-ésima linha da matriz B e o i-ésimo elemento do vetor $\mu \in \Re^{m_2+1}$ respectivamente e I_j e ν_j são a j-ésima coluna da matriz identidade I e o j-ésimo elemento do vetor $\nu \in \Re^{n_1+n_2}$ respectivamente. ∎

Fazendo-se então uso do vetor $\tilde{c}$, Amouzegar e Moshirvaziri [2] desenvolvem um algoritmo de penalidade para resolver uma sequência de problemas do tipo:

$$
\begin{array}{llll}
P(s^i, M) & \min & \tilde{c}^T z + M{s^i}^T z \\
 & \text{s.a} & z \in \bar{Z}
\end{array}
$$

onde o vetor $s^i \in S$ é atualizado e o parâmetro $M > 0$ é incrementado ao longo das iterações do algoritmo descrito em [2, pg.261].

4.2 Algoritmo *branch-and-bound* modificado

Recorde-se que, no algoritmo de Hansen *et al.* [29], em cada nó da árvore de enumeração é resolvido o problema relaxado (RPLDN), obtendo-se uma solução $(\bar{x}, \bar{y})$. Caso este ponto fosse viável para (PLDN) ou $f_1(\bar{x}, \bar{y}) \leq f_{opt}$, onde f_{opt} é o valor na função objetivo do líder do melhor ponto viável para (PLDN) conhecido até o momento, então o subproblema associado a este nó não precisaria mais ser particionado.

Ao longo de um processo de enumeração, quanto mais rápido bons pontos viáveis para (PLDN) forem encontrados, mais rápido os limites inferiores (f_{opt}) serão atualizados e sendo assim menos nós da árvore serão explorados, acelerando então a convergência do algoritmo.

A modificação introduzida neste algoritmo se propõe exatamente a melhorar estes limites inferiores com a adição de um algoritmo específico ao algoritmo de Hansen *et al.*. Recorde-se então o Algoritmo 1 de equilíbrio de Campêlo (pg.36) utilizado para determinar um ponto de equilíbrio do problema $P(M)$ na ausência de restrições do primeiro nível.

Partindo-se de um ponto $z_0 \in Z$, o Algoritmo 1 encontra um ponto de equilíbrio $(\bar{z}, \bar{s}) \in Z \times S$, resolvendo-se apenas dois problemas lineares. Além disso, note que dentre todos os pontos $z \in Z$ satisfazendo $\bar{s}^T z = 0$, o ponto $\bar{z}$ é aquele que fornece o maior valor da função objetivo do líder, em outras palavras, $\bar{z}$ é o melhor ponto viável para PLDNP na face $\{z \in Z : \bar{s}^T z = 0\}$.

Vale a pena mencionar que a idéia de se utilizar um algoritmo específico para se determinar bons limites inferiores em um algoritmo *branch-and-bound* também é utilizada por Audet *et al.* [5]. No entanto, o algoritmo de Audet *et al.* é voltado a resolução global do problema de programação bilinear e não se conhece na literatura o uso de um algoritmo específico que melhore os limites inferiores de um algoritmo *branch-and-bound* voltado para a resolução de (PLDNP).

Fazendo-se uso do Algoritmo 1 de equilíbrio de Campêlo e do algoritmo *branch-and-bound* de Hansen *et al.*, é proposto então um novo algoritmo para se resolver globalmente o problema (PLDNP). Este algoritmo, na verdade, é o algoritmo de Hansen *et al.* onde os Passos 4 e 5 (pg.54) são substituídos pelo Algoritmo 1 de

equilíbrio, como mostrado a seguir.

Passo 4.1 Seja $(\bar{x}, \bar{y})$ a solução do problema (RPLDN'). Encontre então o ponto $(\bar{x}, \dot{y}) \in \Re^{n_1} \times \Re^{n_2}$, como indicado no **item 3** (pg.48) e tome ${z_0}^T = [\bar{x}^T \ \dot{y}^T \ \bar{w}^T] \in Z$, onde $\bar{w} = a - A_1\bar{x} - A_2\dot{y}$;

Passo 4.2 Resolva o problema $\max\{c^T z_0 - M{z_0}^T s : s \in S\}$ pelo método simplex, obtendo uma solução $\hat{s}$;

Passo 5.1 Resolva o problema $\max\{c^T z - Mz^T\hat{s} : z \in Z\}$ pelo método simplex *big-M*, obtendo uma solução $\hat{z}^T = [\hat{x}^T \ \hat{y}^T \ \hat{w}^T]$, que por sua vez é viável para (PLDNP). Se $f_{opt} < c^T\hat{z}$, faça $f_{opt} = c^T\hat{z}$ e $(x^\star, y^\star) = (\hat{x}, \hat{y})$;

Passo 5.2 Se ${c_1}^T\bar{x} + {c_2}^T\bar{y} \leq f_{opt}$ então vá para o Passo 13 (backtracking).

Estes novos passos compreendem o Algoritmo 1 de equilíbrio de Campêlo. Note ainda que tal como os Passos 4 e 5 originais, os novos passos resolvem apenas dois problemas lineares; portanto nenhum esforço computacional adicional foi acrescentado.

Uma característica importante deste novo método diz respeito ao aspecto computacional. Perceba que os quadros ótimos de Z e S são armazenados e que em cada nó é necessário apenas modificar as funções objetivos dos problemas lineares através de métodos de pós-otimização.

A obtenção de bons limites inferiores, com o uso do Algoritmo 1 de equilíbrio, é claramente visível. No próximo capítulo será apresentado um estudo computacional detalhado onde é possível verificar a melhora ocorrida tanto no algoritmo *branch-and-bound*, como no algoritmo de penalidade de Campêlo.

Capítulo 5

Resultados Computacionais

Este capítulo tem por objetivo mostrar os resultados computacionais obtidos ao se testar os algoritmos de Campêlo [16], Hansen *et al.* [29] e os algoritmos de penalidade e *branch-and-bound* modificados apresentados no capítulo anterior.

Os problemas testes utilizados foram obtidos através do gerador de problemas aleatórios de Calamai e Vicente [14]. O método utilizado pelos autores é dividido em 2 partes. Na primeira delas são gerados p subproblemas lineares em dois níveis, onde $p = \min\{n_1, n_2\}$ e n_1 e n_2 são o número de variáveis atribuídas ao líder e ao seguidor, respectivamente. Cada subproblema gerado possui duas variáveis (uma delas atribuída ao líder e outra ao seguidor) e quatro restrições.

Estes subproblemas podem ser de 5 tipos. Os três primeiros tipos possuem apenas ótimos globais e os dois últimos possuem um ótimo local e um global. Na notação utilizada p_i representa o número de subproblemas do tipo i contidos no total de subproblemas p.

Após gerados, os subproblemas são agrupados formando um (PLDNP) separável com $n_{12} = n_1 + n_2$ variáveis, $m = 4p - p'$ restrições, onde $p' = \max\{n_1, n_2\} - p$, 2^{p_2} ótimos globais e $2^{p_2+p_4+p_5} - 2^{p_2}$ ótimos locais não globais.

A segunda etapa elimina a separabilidade do (PLDNP) gerado através da seguinte transformação de variáveis: $\left[x'^T\ y'^T\right]^T = HD^{-1}H \cdot \left[x^T\ y^T\right]^T$, onde $D = \text{diag}(d_{xy})$ e $H = \begin{bmatrix} I_x - 2v_x v_x^T & 0 \\ 0 & I_y - 2v_y v_y^T \end{bmatrix}$.

Nesta transformação, H é uma matriz bloco-diagonal formada por submatrizes de Householder, I_x e I_y são matrizes identidade de ordens n_1, n_2, n_{12} respectivamente, e $v_x \in \Re^{n_1}$, $v_y \in \Re^{n_2}$, $d_{xy} \in \Re^{n_{12}}$ são vetores gerados aleatoriamente com $v_x^T v_x = v_y^T v_y = 1$ e $d_{xy} > 0$.

A densidade e o condicionamento da matriz de restrições são influenciadas pelos parâmetros τ e δ respectivamente, sendo que τ determina o número de elementos não nulos no vetor $\left[{v_x}^T\ {v_y}^T\right]$ e 10^δ é o número de condicionamento da matriz D na norma euclidiana ($\|\cdot\|_2$).

Certas características, no entanto, são indesejáveis neste gerador como por exemplo o número de restrições que é relativamente grande se comparado ao número de variáveis. Uma outra característica é que os problemas gerados possuem variáveis livres ao invés de possuírem variáveis não negativas como é usual nos problemas lineares.

Em [16, pg.125], Campêlo introduz algumas modificações neste gerador baseado em sugestões de seus autores. A primeira delas consiste em se eliminar restrições redundantes dos problemas gerados reduzindo-se então o número de restrições para $m = p_1 + 2p_3 + 3(p_2 + p_4 + p_5) + 1$. A segunda consiste em se adicionar restrições de não negatividade aos problemas. Esta última modificação, apesar de não alterar o ótimo global do problema, pode eliminar alguns ótimos locais ou mesmos gerar outros.

Foram obtidos aleatoriamente, utilizando-se o gerador de Calamai e Vicente [14], disponível na Internet, e adotando-se as modificações de Campêlo [16], um total de 384 problemas testes com as seguintes características:

1. O número total de variáveis ($n = n_1 + n_2$) variou de 50 a 200, sendo que em algums problemas foram atribuídas 25% destas variáveis ao líder e em outros 75%. Ressalta-se que uma proporção semelhante de variáveis atribuídas ao líder e ao seguidor é encontrada nos experimentos de Bard e Moore [6];

2. O número de restrições m ficou em torno de 30% do número total de variáveis e nenhuma restrição foi atribuída ao problema do primeiro nível;

3. Para um dado número de variáveis, foram considerados três números de restrições, gerando três problemas diferentes, sendo que o i-ésimo problema ($i = 1, 2, 3$) é relacionado ao i-ésimo valor de m.

4. O parâmetro δ, que controla o número de condicionamento da matriz D, foi fixado a 1;

5. O parâmetro τ, que influencia a densidade da matriz de restrições, foi fixado em 10%, 40%, 60% e 80% do número total de variáveis $n = n_1 + n_2$.

Os problemas gerados foram divididos em quatro grupos distintos de acordo com os valores τ. Cada grupo é composto por um total de 96 problemas. Na tabela 5.1 abaixo, (μ_d) é a densidade média, (σ_d) é o desvio padrão, $(\min_d)$ e $(\max_d)$ são as densidades mínima e máxima das matrizes de restrição de cada grupo.

τ (%)	μ_d (%)	σ_d (%)	$\min_d$ (%)	$\max_d$ (%)
10	4.5	1.45	2.3	8.4
40	18.8	2.10	16	26
60	41.0	2.4	33.0	45.5
80	63.5	3.5	50.0	72.0

Tabela 5.1: Estatísca das densidades de cada grupo de problemas

Os algoritmos estudados foram implementados em DELPHI, com excessão do algoritmo original de penalidade de Campêlo que foi implementado em C. Todos algoritmos utilizam o mesmo código do algoritmo Simplex utilizado para resolver os subproblemas lineares. Ressalta-se que os algoritmos foram executados em um Pentium II 333 MHz.

Os algoritmos de penalidade original e modificado utilizaram para o parâmetro ε um valor de 0.013. Para os algoritmos *branch-and-bound* original e modificado foi utilizado uma heurística proposta por Hansen *et al.* [29] para se determinar uma solução viável inicial para (PLDNP). Ressalta-se também que os algoritmos *branch-and-bound* e o método enumerativo do algoritmo de penalidade modificado utiliza a regra *depth-first* (pg.24) como método de exploração dos nós da árvore enumerativa.

Os quatros algoritmos apresentados neste trabalho foram testados para o conjunto de problemas descritos anteriormente. Os resultados computacionais para o algoritmo de penalidade original de Campêlo [16] são apresentados na Tabela 5a e para os algoritmos de penalidade modificado, *branch-and-bound* original e *branch-and-bound* modificado, são apresentados nas Tabelas 5b, 5c, 5d e 5e. Estas últimas tabelas são divididas de acordo com os valores das densidades dos problemas testes influenciadas pelo parâmetro τ como descrito na tabela 5.1.

A Tabela 5a mostra o desempenho do algoritmo de penalidade original. O símbolo (*) indica que um dado problema não pode ser resolvido em um tempo

limite de 900s. Observa-se que de um total de 384 problemas testes, o algoritmo não conseguiu resolver 113 problemas, o que representa 30% do total. A ineficiência observada neste algoritmo é decorrente do método *outer approximation* adaptado que gera um número excessivo de vértices nas iterações do algoritmo global.

As Tabelas 5b a 5e são organizadas como se segue. Cada linha destas tabelas contém um número de 3 problemas testes com o mesmo número de váriáveis atribuídas ao líder e ao seguidor. O i-ésimo problema ($i = 1, 2, 3$) está relacionado a columa m_i, T_i, N_i e S_i que representam respectivamente o número de restrições de cada problema, o tempo gasto pelos três algoritmos para encontrar um ótimo global, o número de nós e de subproblemas gerados pelos algoritmos *branch-and-bound*.

Ressalte-se que o número de subproblemas gerados fornece uma melhor idéia do esforço computacional realizado pelos algoritmos *branch-and-bound* do que o número de nós. Isto se deve ao fato de que em um mesmo nó podem ser gerados mais de um subproblema como explicado anteriormente no algoritmo de Hansen *et al.* [29].

A Tabela 5.2 apresenta um resumo dos tempos gastos mostrados nas Tabelas 5b a 5e. Para cada grupo de problemas e para cada algoritmo, com excessão do algoritmo de penalidade original, são fornecidos o tempo médio (μ_T) e o desvio padrão (σ_T). Observando-se esta tabela, as seguintes conclusões podem ser inferidas:

1. O desempenho médio do algoritmo *branch-and-bound* original é superior ao do algoritmo de penalidade modificado. No entanto, observa-se que esta diferença diminui a medida que a densidade dos problemas considerados aumentam;

2. A modificação introduzida no algoritmo *branch-and-bound* original resultou em um expressivo melhoramento em seu desempenho, e esta melhora mostrou-se maior à medida que a densidade dos problemas considerados aumenta;

3. O algoritmo *branch-and-bound* modificado mostrou ser significantemente mais eficiente que o algoritmo de penalidade modificado e esta diferença é levemente afetada pela densidade dos problemas;

4. O desvio padrão aumenta à medida que a densidade aumenta para o algoritmo *branch-and-bound* original. No entanto, a variabilidade permanece quase que inalterada para o algoritmo de penalidade modificado;

5. A modificação introduzida no algoritmo *branch-and-bound* original levou a uma redução do desvio padrão deste algoritmo. Este resultado mostra que o algoritmo 1 de equilíbrio quando introduzido no método *branch-and-bound* torna o mesmo mais robusto.

	Penalidade Modificado		*Branch-and-Bound* Original		*Branch-and-Bound* Modificado	
τ (%)	μ_T (s)	σ_T (s)	μ_T (s)	σ_T (s)	μ_T (s)	σ_T (s)
10	3.29	5.23	2.53	2.66	2.30	2.52
40	3.01	5.16	2.56	2.82	2.01	2.56
60	3.51	5.29	2.82	3.83	2.07	2.48
80	3.63	4.56	3.74	5.99	2.42	3.01

Tabela 5.2: Estatísticas do tempo médio gasto pelos grupos de problemas

Finalmente é mostrada a porcentagem de problemas em que um certo algoritmo supera o outro no conjunto de problemas testes. O algoritmo *branch-and-bound* original superou ou foi comparável (diferença de tempo menor que 0.05s) ao algoritmo de penalidade modificado em 77% dos 384 problemas. O algoritmo *branch-and-bound* modificado superou ou foi comparável ao original em 76% dos problemas testes.

Apesar da visível eficiência do algoritmo *branch-and-bound* original comparativamente ao algoritmo de penalidade modificado, é importante lembrar que este último é capaz de resolver problemas que não satisfaçam as hipóteses de compacidade assumidas pelo primeiro.

Uma outra observação importante refere-se aos algoritmos *branch-and-bound* original e modificado. Perceba que apesar do visível melhoramento no algoritmo original, ainda existe uma pequena porcentagem de problemas em que o algoritmo modificado é menos eficiente que o algoritmo original.

Observe entretanto que nestes problemas o número de subproblemas gerados pelo algoritmo modificado foi, em sua totalidade (com a excessão de 4 problemas), o mesmo número de subproblemas gerados pelo original. Visto que o número de subproblemas gerados são idênticos, a diferença de tempo é devida às dimensões dos problemas lineares considerados. De fato, recorde-se que os Passos 4 e 5 do algoritmo *branch-and-bound* original, que consideram problemas lineares de dimensões menores devido a eliminação de variáveis, foram substituídos pelo Algoritmo 1 de equilíbrio que trabalha com os problemas lineares nas dimensões originais, o que acarreta uma pequena diferença no tempo de resolução.

					$\tau = 10\%$			$\tau = 40\%$			$\tau = 60\%$			$\tau = 80\%$		
n	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3
50	12	15	17	19	0.06	0.16	0.77	0.33	1.81	24.44	0.05	0.22	0	0	0.11	0.11
50	38	15	17	19	0.66	3.35	3.84	0.05	3.79	*	0.11	0.61	10.76	0.98	1.15	2.31
60	15	18	20	22	0.11	0.27	0.71	0.49	0.27	0.77	0	0.5	0.39	0	0.05	3.02
60	45	18	20	22	1.21	8.9	5.27	12.91	0.5	*	0.06	6.37	2.8	0.06	4.51	4.17
70	17	20	22	24	0.11	0.33	0.99	0	1.15	16.76	0.11	1.86	0.55	0	1.76	2.75
70	53	20	22	24	2.59	1.81	8.4	0.11	8.51	*	0.11	*	*	0.38	3.68	*
80	20	23	25	27	0.11	0.39	1.32	0	0.44	*	0.11	0.17	0.61	0.06	0	0.22
80	60	23	25	27	3.24	2.91	8.07	0	0.22	*	0.17	1.76	*	0.33	6.26	2.31
90	22	25	27	29	0.11	0.93	1.59	0.05	0.77	27.84	0	*	0.27	0.06	1.76	*
90	68	25	27	29	5.82	2.52	15.16	0.06	*	*	0.22	3.95	*	2.36	2.42	*
100	25	28	20	32	0.17	11.92	*	0.17	1.1	0.72	0.32	1.04	0.38	0.33	0.5	*
100	75	28	20	32	2.58	25.21	*	0.27	0.88	*	0.05	*	21.36	0.27	4.23	*
110	27	30	32	34	0.16	14.99	4.01	0.16	9.23	6.04	0.22	0.06	0.28	0.05	0	0.38
110	83	30	32	34	12.58	39.22	*	13.13	*	*	0.33	*	*	0.11	1.53	*
120	30	33	35	37	0.27	19.06	*	*	1.92	0.11	1.37	1.21	*	0.06	0.11	*
120	90	33	35	37	4.55	175.59	*	0.38	*	27.19	0.39	6.26	*	0.38	8.35	14.5
130	32	35	37	39	0.28	22.19	*	1.1	2.91	0.33	0.6	14	*	0.06	0.06	33.12
130	98	35	37	39	0.55	*	*	54.98	*	*	0.44	0.55	1.2	0.5	*	*
140	35	38	40	42	0.33	15.93	*	0.33	3.68	*	0.77	1.04	*	0.28	*	*
140	105	38	40	42	150	*	*	0.55	*	*	0.55	*	*	0.17	93.43	217.4
150	37	40	42	44	0.6	151.65	*	*	0.44	134.57	0.06	*	*	0.06	1.54	44.05
150	113	40	42	44	28.51	*	*	0.6	16.59	*	2.09	128.8	*	1.26	*	*
160	40	43	45	47	0.49	*	8.63	1.21	1.76	6.59	*	5.05	11.15	1.04	2.64	*
160	120	43	45	47	34.99	*	*	0.22	1.43	11.92	4.45	*	34	0.17	*	35.59
170	42	45	47	49	0.55	*	*	0.49	4.01	7.09	0.06	*	*	9.33	*	*
170	128	45	47	49	*	346.09	*	0.71	2.96	*	0.82	*	*	3.35	*	2.86
180	45	48	50	52	0.77	*	*	0.06	*	0.27	0.11	*	0.82	0.11	*	5.49
180	135	48	50	52	*	*	*	0.94	46.68	7.08	0.93	50.31	*	0.83	132.42	68.93
190	47	50	52	54	1.54	*	*	0.82	*	2.47	0.11	2.41	0.17	0.11	1.1	0.16
190	143	50	52	54	531.46	*	*	1.09	18.51	17.08	1.04	135.72	*	0.99	*	*
200	50	53	55	57	0.72	*	*	1.6	5.11	0.28	*	*	0.93	1.82	*	2.7
200	150	53	55	57	*	*	*	10.54	*	*	1.16	*	*	1.1	*	*

Tabela 5a: Resultados computacionais para o algoritmo de penalidade original

					P. Modificado			B&B Original									B&B Modificado								
n	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.06	0.05	0.11	0.00	0	12	0.05	2	13	0.06	6	18	0.05	0	12	0.06	2	13	0.05	6	15
50	38	15	17	19	0.05	0.05	0.28	0.05	2	12	0.06	2	14	0.11	6	19	0.06	0	11	0.11	2	14	0.05	6	17
60	15	18	20	22	0.05	0.11	0.27	0.05	0	13	0.05	2	16	0.11	6	21	0.06	0	13	0.11	2	15	0.11	6	18
60	45	18	20	22	0.11	0.27	0.33	0.05	2	13	0.17	2	17	0.16	8	23	0.11	0	12	0.11	2	15	0.17	2	17
70	17	20	22	24	0.11	0.16	0.49	0.11	2	16	0.11	2	16	0.16	6	23	0.11	0	15	0.11	2	15	0.17	6	20
70	53	20	22	24	1.04	0.22	0.71	0.22	2	17	0.21	2	19	0.28	6	24	0.17	0	16	0.17	2	17	0.22	2	20
80	20	23	25	27	0.11	0.27	0.5	0.17	0	18	0.22	2	20	0.27	6	26	0.22	0	18	0.22	2	18	0.33	6	25
80	60	23	25	27	0.22	0.22	0.55	0.28	2	18	0.33	2	22	0.38	4	20	0.33	0	17	0.33	2	20	0.33	2	17
90	22	25	27	29	0.17	0.33	0.49	0.33	0	22	0.39	4	25	0.44	6	28	0.33	0	22	0.33	2	24	0.44	6	28
90	68	25	27	29	0.33	3.57	1.81	0.44	2	22	0.55	2	22	0.66	6	28	0.44	0	21	0.49	2	21	0.55	6	25
100	25	28	20	32	0.28	0.77	0.77	0.55	2	26	0.66	6	29	0.61	2	25	0.44	0	25	0.55	6	29	0.55	2	25
100	75	28	20	32	0.71	1.93	1.26	0.82	4	30	0.82	6	29	0.76	2	21	0.72	0	25	0.82	6	29	0.28	0	1
110	27	30	32	34	0.39	1.21	0.99	0.66	2	26	0.82	6	29	0.82	2	27	0.66	0	25	0.77	2	27	0.77	2	27
110	83	30	32	34	0.60	2.25	2.04	1.05	2	27	1.21	6	31	1.32	8	30	1.04	2	27	1.04	2	29	1.26	8	30
120	30	33	35	37	0.44	1.10	0.82	0.99	0	28	1.20	6	32	1.15	2	30	0.93	0	28	1.05	2	30	1.1	2	30
120	90	33	35	37	2.20	3.30	15.93	1.65	6	34	1.70	6	34	1.81	8	33	0.60	0	5	1.54	2	32	1.76	8	33
130	32	35	37	39	0.43	1.32	0.88	1.32	0	32	1.48	6	33	1.54	2	33	1.27	0	32	1.37	2	31	1.37	2	33
130	98	35	37	39	1.04	4.12	21.37	2.03	2	32	2.14	6	36	2.58	14	41	1.87	0	31	2.09	6	36	2.25	10	37
140	35	38	40	42	0.55	1.76	1.37	1.76	0	33	2.03	6	36	1.92	2	33	1.59	0	33	1.76	2	34	1.82	2	33
140	105	38	40	42	1.26	6.15	3.08	2.69	2	36	2.91	6	39	3.08	8	37	2.48	0	35	2.64	2	37	2.96	8	37
150	37	40	42	44	0.77	2.36	1.87	2.14	2	34	2.41	6	38	2.47	2	36	1.92	0	33	2.31	6	38	2.25	2	36
150	113	40	42	44	28.95	7.58	26.2	3.19	2	35	3.57	6	41	4.01	14	43	3.07	2	35	3.35	6	41	3.85	14	43
160	40	43	45	47	0.83	5.22	1.98	2.81	2	37	3.02	6	40	3.13	2	38	2.52	0	36	2.80	6	40	2.91	2	38
160	120	43	45	47	2.48	8.79	5.44	4.06	2	38	4.62	6	44	4.34	2	35	3.84	0	37	4.17	2	42	4.01	2	34
170	42	45	47	49	0.94	2.09	2.75	3.46	2	40	3.51	4	40	3.52	2	36	3.08	0	39	3.18	2	39	3.29	2	37
170	128	45	47	49	2.14	4.28	4.83	5.33	2	43	5.49	2	44	5.76	8	43	4.94	0	42	5.00	2	42	5.61	8	43
180	45	48	50	52	1.21	5.61	3.35	4.40	0	42	4.56	4	42	4.72	2	42	4.01	0	42	4.17	2	41	4.29	2	41
180	135	48	50	52	2.59	13.40	6.65	6.48	2	43	7.03	6	49	7.03	2	41	6.09	0	42	6.59	6	49	6.7	2	41
190	47	50	52	54	1.59	6.15	4.23	5.10	2	43	5.39	6	45	5.43	2	42	4.51	0	42	4.94	6	45	4.94	2	42
190	143	50	52	54	8.40	7.47	17.14	7.91	6	47	8.13	2	49	9.22	6	49	7.36	6	47	7.64	2	49	9.45	6	46
200	50	53	55	57	2.30	6.04	4.94	6.54	2	46	6.70	4	46	6.54	2	43	5.88	0	45	1.26	0	5	6.1	2	44
200	150	53	55	57	3.46	8.68	9.5	9.67	2	51	10.1	2	52	10.4	2	52	9.01	0	50	9.45	2	52	11.09	2	52

Tabela 5b: Resultados computacionais para problemas com $\tau = 10\%$

					P. Modificado			B&B Original									B&B Modificado								
n	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.00	0.17	0.11	0.06	2	10	0.05	2	7	0.05	2	9	0.00	0	0	0.00	0	6	0.06	2	9
50	38	15	17	19	0.05	0.11	0.39	0.11	6	13	0.11	6	13	0.82	92	180	0.06	6	13	0.05	6	13	0.66	82	153
60	15	18	20	22	0.06	0.06	0.17	0.06	0	10	0.05	2	10	0.11	2	11	0.05	0	10	0.00	0	0	0.05	0	10
60	45	18	20	22	0.17	0.11	0.22	0.11	2	11	0.16	6	17	0.16	8	16	0.06	0	10	0.17	6	17	0.17	8	16
70	17	20	22	24	0.06	0.11	0.49	0.06	2	10	0.11	2	12	0.11	4	12	0.05	0	1	0.11	0	11	0.11	2	10
70	53	20	22	24	0.22	1.70	0.83	0.22	6	18	0.44	24	45	2.58	308	324	0.22	6	18	0.50	24	45	0.66	48	58
80	20	23	25	27	0.11	0.17	0.44	0.17	2	12	0.17	2	12	0.17	4	12	0.00	0	1	0.16	0	11	0.16	2	10
80	60	23	25	27	0.38	0.49	6.04	0.33	6	16	1.16	80	92	2.58	200	209	0.33	6	16	0.33	6	18	2.47	194	203
90	22	25	27	29	0.11	0.28	0.49	0.22	2	13	0.28	2	15	0.33	6	16	0.05	0	1	0.27	0	14	0.28	2	13
90	68	25	27	29	0.44	0.55	6.76	0.66	12	25	0.55	6	18	2.63	138	151	0.50	6	19	0.55	6	18	2.64	132	145
100	25	28	20	32	0.22	0.66	0.65	0.39	2	17	0.38	2	15	0.50	4	17	0.38	0	16	0.11	0	1	0.11	0	1
100	75	28	20	32	0.66	1.10	5.50	0.77	6	21	0.99	6	23	4.23	150	165	0.77	6	21	0.94	6	23	3.46	122	137
110	27	30	32	34	0.49	0.99	0.38	0.50	0	16	0.71	10	25	0.66	2	17	0.44	0	16	0.77	10	25	0.11	0	1
110	83	30	32	34	0.71	21.20	2.80	0.88	2	16	1.48	14	31	1.59	14	29	0.88	2	16	1.48	14	31	1.60	14	29
120	30	33	35	37	0.71	1.05	0.28	0.83	2	19	0.72	2	18	0.77	2	16	0.33	0	5	0.66	0	17	0.11	0	0
120	90	33	35	37	0.66	2.20	2.59	1.49	6	23	1.70	6	23	2.31	18	34	1.37	6	23	1.65	6	23	1.53	6	22
130	32	35	37	39	0.50	1.15	0.33	1.04	2	21	1.10	2	22	1.04	2	19	0.88	0	20	0.99	0	21	0.22	0	1
130	98	35	37	39	1.76	1.21	10.33	1.65	2	20	2.03	6	26	2.75	14	34	1.59	2	20	1.98	6	26	2.74	14	34
140	35	38	40	42	0.60	0.77	1.65	1.32	0	22	1.43	2	23	1.43	2	22	1.26	0	22	0.22	0	0	1.26	0	21
140	105	38	40	42	0.99	2.69	31.41	2.75	6	27	3.46	12	33	3.74	14	34	2.63	6	27	2.75	6	27	3.57	12	33
150	37	40	42	44	2.04	0.49	2.64	0.49	0	4	1.81	2	25	2.37	12	33	0.49	0	4	0.22	0	0	2.19	8	31
150	113	40	42	44	1.98	4.12	5.55	3.13	6	26	5.44	20	40	7.41	38	57	3.08	6	26	3.30	6	26	4.78	18	37
160	40	43	45	47	2.25	1.81	2.03	2.03	0	24	2.36	2	27	2.26	2	24	1.93	0	24	2.20	2	27	0.38	0	1
160	120	43	45	47	2.42	3.68	4.06	4.12	6	26	3.96	2	22	6.42	24	44	3.84	6	26	0.82	0	0	4.12	6	26
170	42	45	47	49	1.04	2.20	2.14	2.36	0	23	2.85	2	29	2.80	2	26	2.20	0	23	2.59	2	29	0.33	0	0
170	128	45	47	49	1.70	22.13	2.74	5.05	6	29	7.75	16	40	5.43	6	29	4.83	6	29	7.03	16	40	5.28	6	29
180	45	48	50	52	0.77	2.30	1.48	0.33	0	0	3.51	2	28	3.35	2	26	0.38	0	0	0.39	0	0	0.55	0	1
180	135	48	50	52	3.46	6.98	10.05	6.81	6	33	7.52	12	35	9.56	20	46	6.43	6	33	7.20	12	35	7.19	6	33
190	47	50	52	54	1.37	3.25	3.29	4.06	0	31	3.90	2	28	4.56	4	32	3.85	0	31	3.62	2	28	0.72	0	2
190	143	50	52	54	5.55	14.39	3.74	7.31	6	31	8.84	12	40	7.86	6	31	6.81	6	31	8.40	12	40	7.52	6	31
200	50	53	55	57	2.69	3.57	1.92	4.83	0	31	5.06	2	32	5.21	2	31	4.51	0	31	4.61	0	31	0.77	0	1
200	150	53	55	57	11.81	8.40	18.78	10.6	8	41	11.8	14	45	11.3	14	38	10.27	8	41	10.99	14	45	9.56	6	31

Tabela 5c: Resultados computacionais para problemas com $\tau = 40\%$

					P. Modificado			B&B Original									B&B Modificado								
n	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.05	0.11	0.05	0.00	2	4	0.05	2	5	0.06	2	3	0.00	0	0	0.00	0	4	0.00	0	0
50	38	15	17	19	0.05	0.17	1.16	0.11	12	17	0.55	142	147	0.83	164	178	0.06	6	11	0.16	16	21	0.93	164	178
60	15	18	20	22	0.06	0.05	0.05	0.00	0	0	0.05	2	9	0.06	2	6	0.00	0	0	0.06	0	8	0.00	0	1
60	45	18	20	22	0.16	0.55	0.44	0.11	6	12	0.27	22	27	1.82	288	294	0.11	6	12	0.17	10	15	0.27	26	32
70	17	20	22	24	0.06	0.11	0.17	0.00	0	0	0.06	2	11	0.11	2	8	0.05	0	0	0.11	0	10	0.00	0	0
70	53	20	22	24	0.22	0.44	0.61	0.22	6	15	0.22	6	13	0.66	36	43	0.22	6	15	0.22	6	13	0.55	30	37
80	20	23	25	27	0.28	0.11	0.28	0.11	2	8	0.05	0	1	0.11	2	8	0.11	0	7	0.00	0	0	0.11	0	2
80	60	23	25	27	0.33	0.50	1.87	0.33	6	14	2.75	202	208	0.99	42	50	0.33	6	14	0.38	10	16	0.99	42	50
90	22	25	27	29	0.11	0.16	0.28	0.00	0	0	0.22	2	11	0.22	2	10	0.05	0	0	0.16	2	11	0.11	0	1
90	68	25	27	29	0.55	0.66	8.52	0.60	12	21	0.60	12	20	1.48	38	46	0.44	6	15	0.50	6	14	1.54	38	46
100	25	28	20	32	0.44	0.61	0.55	0.22	0	8	0.33	2	11	0.38	2	13	0.28	0	8	0.33	2	11	0.11	0	0
100	75	28	20	32	0.54	2.31	6.10	0.88	12	20	1.43	22	29	6.81	208	215	0.61	6	14	1.26	20	27	6.48	208	215
110	27	30	32	34	0.49	0.50	0.33	0.38	0	11	0.11	0	0	0.44	2	11	0.33	0	11	0.11	0	0	0.11	0	0
110	83	30	32	34	0.71	5.00	3.74	0.99	6	15	6.43	156	166	7.14	132	141	0.94	6	15	3.57	62	72	2.25	28	37
120	30	33	35	37	0.93	0.93	0.66	0.55	2	11	0.66	2	14	0.66	2	13	0.11	0	0	0.22	0	3	0.61	0	12
120	90	33	35	37	0.93	1.59	1.97	1.32	6	15	1.81	12	21	1.37	6	14	1.21	6	15	1.76	12	21	1.38	6	14
130	32	35	37	39	1.26	1.54	11.97	0.72	0	14	0.77	2	14	1.10	10	21	0.66	0	14	0.71	0	13	0.88	6	17
130	98	35	37	39	1.48	1.59	1.81	1.87	6	18	1.81	6	17	1.81	8	17	1.81	6	18	1.76	6	17	1.82	6	16
140	35	38	40	42	0.60	0.93	3.84	1.04	2	17	0.99	2	15	1.48	8	23	0.99	0	16	0.94	2	15	1.32	6	20
140	105	38	40	42	2.58	2.20	7.03	2.36	6	19	3.29	14	26	24.0	290	302	2.25	6	19	2.97	14	26	2.91	12	24
150	37	40	42	44	0.44	0.71	2.37	0.16	0	0	1.37	2	18	1.54	4	19	0.22	0	0	1.32	0	17	1.48	4	19
150	113	40	42	44	2.47	9.72	6.97	3.07	6	20	4.72	20	35	2.97	6	19	2.97	6	20	3.73	14	27	2.86	6	19
160	40	43	45	47	3.40	1.42	2.31	2.09	10	21	1.71	2	19	1.54	2	15	2.03	10	21	0.33	0	0	0.38	0	0
160	120	43	45	47	9.40	32.19	2.96	4.23	6	25	4.06	6	21	3.79	6	19	4.06	6	25	3.90	6	21	3.79	6	19
170	42	45	47	49	1.26	7.80	3.29	0.49	0	2	2.03	2	14	2.91	14	25	0.50	0	2	0.94	0	4	2.86	14	25
170	128	45	47	49	3.74	23.78	13.79	4.67	6	24	7.31	18	36	9.39	36	48	4.44	6	24	6.97	18	36	28.45	226	241
180	45	48	50	52	0.83	2.03	2.36	0.33	0	0	2.64	2	21	2.14	2	14	0.38	0	0	2.47	0	20	0.44	0	0
180	135	48	50	52	2.03	10.28	5.83	6.37	6	26	7.69	16	32	9.23	20	39	6.10	6	26	6.26	10	26	6.64	6	26
190	47	50	52	54	0.93	3.52	2.52	0.33	0	0	3.07	2	20	4.06	12	27	0.44	0	0	2.80	0	19	3.85	12	27
190	143	50	52	54	5.11	9.89	6.76	6.32	6	22	8.73	12	31	15.0	44	60	6.04	6	22	8.40	12	31	7.63	12	28
200	50	53	55	57	9.39	1.70	5.66	3.29	2	19	3.79	2	22	3.62	2	19	2.97	0	18	0.55	0	0	12.79	64	85
200	150	53	55	57	2.81	18.45	13.95	7.80	6	23	10.6	14	31	12.9	24	40	7.19	6	23	10.27	14	31	7.75	6	22

Tabela 5d: Resultados computacionais para problemas com $\tau = 60\%$

					P. Modificado			B&B Original									B&B Modificado								
n	n_2	m_1	m_2	m_3	T_1	T_2	T_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3	T_1	N_1	S_1	T_2	N_2	S_2	T_3	N_3	S_3
50	12	15	17	19	0.06	0.05	0.06	0.00	0	0	0.05	10	11	0.06	12	14	0.00	0	0	0.06	10	11	0.05	12	14
50	38	15	17	19	0.22	0.11	0.17	0.11	18	22	0.05	8	10	0.05	2	6	0.16	18	22	0.06	6	9	0.11	2	6
60	15	18	20	22	0.06	0.06	0.28	0.05	0	0	0.06	2	8	0.06	2	5	0.00	0	0	0.05	0	7	0.00	2	5
60	45	18	20	22	0.88	0.88	1.10	0.82	138	142	0.61	78	82	1.43	228	232	0.83	138	142	0.71	78	82	1.26	190	194
70	17	20	22	24	0.06	0.11	0.50	0.05	0	0	0.05	2	6	0.06	2	5	0.00	0	0	0.05	0	5	0.05	0	4
70	53	20	22	24	0.38	0.28	2.91	0.17	6	11	0.33	18	21	5.11	482	486	0.22	6	11	0.22	6	9	2.75	274	278
80	20	23	25	27	0.05	0.16	0.16	0.06	0	0	0.06	0	0	0.11	2	7	0.05	0	0	0.00	0	0	0.06	0	0
80	60	23	25	27	0.38	1.10	0.44	0.28	6	8	0.50	18	21	0.33	8	10	0.22	6	8	0.27	6	9	0.27	8	10
90	22	25	27	29	0.17	0.32	1.48	0.00	0	0	0.16	2	8	0.22	6	11	0.05	0	0	0.17	2	8	0.28	6	11
90	68	25	27	29	0.99	1.21	18.23	0.39	6	10	0.50	8	12	15.4	826	830	0.38	6	10	0.55	8	12	15.11	754	758
100	25	28	20	32	0.33	0.66	1.10	0.22	0	7	0.22	2	8	0.27	2	8	0.22	0	7	0.22	0	7	0.27	2	8
100	75	28	20	32	0.88	0.82	7.14	0.72	6	11	0.55	6	8	1.87	32	37	0.60	6	11	0.61	6	8	1.09	14	19
110	27	30	32	34	0.27	0.33	0.38	0.06	0	0	0.06	0	0	0.22	2	5	0.11	0	0	0.11	0	0	0.11	0	0
110	83	30	32	34	0.99	1.27	9.78	0.88	6	12	5.71	122	126	6.92	140	146	0.82	6	12	0.88	6	11	5.77	122	128
120	30	33	35	37	0.22	0.38	0.61	0.06	0	0	0.11	0	1	0.55	2	11	0.16	0	0	0.16	0	0	0.16	0	1
120	90	33	35	37	1.21	2.69	2.75	1.04	6	10	1.59	12	17	3.30	34	38	1.05	6	10	1.54	12	17	1.70	16	20
130	32	35	37	39	0.33	0.77	1.54	0.11	0	0	0.11	0	0	0.66	6	12	0.17	0	0	0.17	0	0	0.55	2	8
130	98	35	37	39	1.53	2.25	17.46	1.48	6	13	1.59	6	12	19.5	238	245	1.49	6	13	1.48	6	12	13.13	178	185
140	35	38	40	42	0.83	2.09	11.98	0.94	8	15	10.7	170	177	0.66	2	8	1.04	8	15	6.48	100	107	0.27	0	0
140	105	38	40	42	1.43	7.36	5.55	1.98	6	14	5.87	34	43	41.8	600	608	1.92	6	14	5.66	34	43	3.02	16	24
150	37	40	42	44	0.66	0.77	3.74	0.16	0	0	1.04	2	12	1.15	6	13	0.17	0	0	0.93	0	11	1.04	4	12
150	113	40	42	44	1.65	2.91	5.33	2.31	6	13	2.69	6	13	6.54	36	43	2.20	6	13	2.42	6	13	4.50	24	31
160	40	43	45	47	1.43	1.32	3.29	0.99	0	9	0.93	2	8	0.99	2	8	0.93	0	9	0.27	0	0	0.88	0	7
160	120	43	45	47	2.14	17.41	2.36	2.81	6	12	3.24	6	15	2.86	6	12	2.63	6	12	2.97	6	15	2.86	6	12
170	42	45	47	49	2.03	2.69	16.20	9.78	98	104	1.48	2	11	7.91	70	76	1.70	8	14	0.66	0	3	5.11	42	48
170	128	45	47	49	7.96	8.57	5.27	8.84	36	45	5.38	12	21	8.62	22	31	8.30	36	45	5.38	12	21	5.33	12	21
180	45	48	50	52	0.82	2.58	3.13	0.33	0	0	1.81	2	12	1.76	2	11	0.38	0	0	1.65	0	11	1.59	0	10
180	135	48	50	52	2.08	10.98	8.63	4.73	6	16	5.38	10	18	11.8	34	43	4.56	6	16	5.00	10	18	5.98	12	21
190	47	50	52	54	0.93	1.70	2.31	0.33	0	0	0.55	0	1	3.13	12	19	0.50	0	0	0.61	0	1	3.18	12	19
190	143	50	52	54	2.41	9.89	14.50	5.27	6	16	7.47	12	24	12.7	34	44	4.95	6	16	7.08	12	24	6.87	12	22
200	50	53	55	57	3.51	10.99	2.25	2.47	0	13	13.2	24	36	2.75	2	14	2.64	0	13	6.65	6	18	2.63	2	14
200	150	53	55	57	2.75	10.38	9.17	6.43	6	17	8.51	12	23	14.3	34	45	6.09	6	17	8.35	12	23	8.08	12	23

Tabela 5e: Resultados computacionais para problemas com $\tau = 80\%$

Conclusões e perspectivas

O objetivo inicial deste trabalho era essencialmente a análise da eficiência computacional do algoritmo de penalidade de Campêlo comparativamente a um outro algoritmo global conhecido na literatura. Em particular foi escolhido o algoritmo *branch-and-bound* de Hansen *et al.* devido aos bons resultados computacionais obtidos pelos autores, mostrando ser este método um dos mais eficientes conhecidos atualmente.

A impossibilidade de obtenção do código do algoritmo de Hansen *et al.* levou a necessidade de sua implementação. A mesma foi realizada utilizando-se o mesmo código do algoritmo simplex utilizado também no código do algoritmo de penalidade de Campêlo.

Devido também a impossibilidade de se usar os mesmos problemas testes de Hansen *et al.*, pois os mesmos não estão disponíveis, foi gerado aleatoriamente um conjunto de instâncias, onde se procurou seguir uma linha semelhante àquela utilizada em outros trabalhos, no que diz respeito ao número de variáveis, restrições e densidade dos problemas.

Os resultados computacionais para os algoritmos de *branch-and-bound* original confirmaram as afirmações de Hansen *et al.* no que diz respeito a eficiência do método. O algoritmo conseguiu resolver todos os problemas testes em um tempo bastante satisfatório. Entretanto o algoritmo de penalidade original não resolveu em um tempo limite de 900s um total de 30% dos problemas. Além disto não foi possível identificar quais parâmetros (número de variáveis atribuídas ao seguidor, densidade, etc) mais afetam o desempenho do mesmo.

Ainda em relação a este algoritmo, um detalhe importante observado foi que sua ineficiência é aparentemente causada pelo algoritmo *outer approximation.* Observou-se que este algoritmo gerava uma quantidade excessiva de vértices, o que consequen-

temente levava a sua visível ineficiência.

A substituição do algoritmo *outer approximation* pelo algoritmo enumerativo proposto, tornou o algoritmo de penalidade competitivo com o algoritmo *branch-and-bound* de Hansen *et al.*. O mesmo passou a resolver os problemas antes não resolvidos em um tempo satisfatório.

A introdução do algoritmo de ponto de equilíbrio nos nós da árvore de enumeração do algoritmo *branch-and-bound* de Hansen *et al.* tornou o mesmo mais robusto e eficiente que o original. Na quase totalidade dos problemas, tanto o número de nós quanto o número de subproblemas gerados foram reduzidos consideravelmente.

Apesar do melhor desempenho do algoritmo *branch-and-bound* original e modificado, em ralação ao algoritmo de penalidade modificado, é importante ressaltar que o número de nós gerados é consequência direta das soluções dos problemas relaxados, o que poderia ser comprometido se estes fossem ilimitados. Em outras palavras, o algoritmo *branch-and-bound* poderia ser ineficiente comparativamente ao algoritmo de penalidade original se as hipótesis de compacidade não fossem assumidas.

Como sugestões para trabalhos futuros, poder-se-ia citar o melhoramento do algoritmo enumerativo. De fato, o desenvolvimento de heurísticas para melhorar as regras de ramificação ou o desenvolvimento de regras de poda mais eficientes poderiam melhorar o desempenho do algoritmo, principalmente nos casos em que não existam soluções satisfazendo a complementaridade e consequentemente necessitando que a árvore seja completamente explorada.

Um outro campo de estudo poderia ser a eliminação das hipóteses de compacidade assumidas pelo algoritmo *branch-and-bound*, tornando o mesmo capaz de identificar problemas ilimitados ou mesmo solucionar àqueles que possuem solução ótima, mas onde o problema relaxado seja ilimitado.

Referências Bibliográficas

[1] Alarie, S., Audet, C., Jaumard, B., Savard, G. "Concavity cuts for disjoint bilinear programming". *Mathematical Programming*, n. 90, pp. 373–398, 2001.

[2] Amouzegar, M., Moshirvazari, K. "A penalty method for linear bilevel programming problems". In Migdalas, A., Pardalos, P., Värbrand, P. (eds), *Multilevel optimization: algorithms and applications*. Nonconvex Optimization and its Applications 20: 251–271, Kluwer Academic Publishers, 1998.

[3] Amouzegar, M., Moshirvazari, K. "Determinig optimal pollution control policies: An application of bilevel programming". *European Journal of Operational Research Society*, v. 119, pp. 100–120, 1999.

[4] Audet, C., Hansen, P., Jaumard, B., Savard, G. "Links between linear bilevel and mixed 0–1 programming problems". *Journal of Optmization: Theory and Applications*, v. 93, n. 2, pp. 273–300, 1997.

[5] Audet, C., Hansen, P., Jaumard, B., Savard, G. "A symmetrical linear maxmin approach to disjoint bilinear programming". *Mathematical Programming*, n. 85, pp. 573–592, 1999.

[6] Bard, J., Moore, J. "A branch and bound algorithm for the bilevel programming problem". *SIAM Journal on Scientific and Statistical Computing*, v. 11, pp. 281–292, 1990.

[7] Bard, J., Plummer, J., Sourie, J. "A bilevel programming approach to determining tax credits for biofuel production". *European Journal of Operational Research Society*, v. 120, pp. 30–46, 2000.

[8] Beale, E. "On quadratic programming". *Naval Research Logistics Quarterly*, v. 6, pp. 227–243, 1959.

[9] Ben-Ayed, O., Blair, C. "Computational difficulties of bilevel linear programming". *Operations Research*, v. 38, pp. 556–560, 1990.

[10] Ben-Ayed, O., Blair, C., Boyce, D., LeBlanc, L. "Construction of a real-world bilevel linear programming model of the highway design problem". *Annals of Operations Research*, v. 34, pp. 219–254, 1992.

[11] Ben-Ayed, O., Boyce, D., Blair, C. "A general bilevel linear programming formulation of the network design problem". *Transportation Research*, v. 22 B, pp. 311–318, 1988.

[12] Benson, H. "On the structure and properties of a linear multilevel programming problem". *Journal of Optimization Theory and Applications*, v. 60, pp. 353–373, 1989.

[13] Bialas, W., Karwan, M. "Two-level linear programming". *Management Science*, v. 30, pp. 1004–1020, 1984.

[14] Calamai, P., Vicente, L. "Generating linear and linear-quadratic bilevel programming problems". *SIAM Journal on Scientific and Statistical Computing*, v. 14, pp. 770–782, 1993.

[15] Campêlo, M., Scheimberg, S. "Theoretical and computational results for a linear bilevel problem". In Hadjisavvas, N., Pardalos, P. (eds), *Advances in Convex Analysis and Global Optimization.* Nonconvex Optimization and its Applications 54: 269–281, Kluwer Academic Publishers, 2001.

[16] Campêlo, M. *Programação linear em dois níveis: uma abordagem teórica e computacional.* PhD thesis, Universidade Federal do Rio de Janeiro, 1999.

[17] Campêlo, M., Dantas, S., Scheimberg, S. "A note on a penalty function approach for solving bilevel linear programs". *Journal of Global Optimization*, v. 16, pp. 245–255, 2000.

[18] Campêlo, M., Scheimberg, S. "A note on a modified simplex approach for solving bilevel linear programming problems". *European Journal of Operational Research Society*, v. 126, pp. 454–458, 2000.

[19] Candler, W., Fortuny-Amat, J., McCarl, B. "The potential role of multilevel programming in agricultural economics". *American Journal of Agricultural Economics*, v. 63, pp. 521–531, 1981.

[20] Candler, W., Townsley, R. "A linear two-level programming problem". *Computers and Operations Research*, v. 9, pp. 59–76, 1982.

[21] Chen, P., Hansen, P., Jaumard, B. "On-line and off-line vertex enumeration by adjacency lists". *Operations Research Letters*, v. 10, n. 7, pp. 403–409, 1991.

[22] Chen, Y., Florian, M. "Congested O-D trip demand adjustment problem: bilevel programming formulation and optimality conditions". In Migdalas, A., Pardalos, P., Värbrand, P. (eds), *Multilevel optimization: algorithms and applications.* Nonconvex Optimization and its Applications 20: 1–22, Kluwer Academic Publishers, 1998.

[23] Clegg, J., Smith, M., Xiang, Y., Yarrow, R. "Bilevel programming applied to optimising urban transportation". *Transportation Research*, v. 35, pp. 41–70, 2001.

[24] Dempe, S., Bard, J. "Bundle trust-region algorithm for bilinear bilevel programming". *Journal of Optmization: Theory and Applications*, v. 110, pp. 265–288, 2001.

[25] Falk, J. "A linear max-min problem". *Mathematical Programming*, v. 5, pp. 169–188, 1973.

[26] Florian, M., Chen, Y. *A bilevel programming approach to estimating O-D matrix by traffic counts.* Technical Report CRT-750, Centre de Recherche sur les Transports, 1991.

[27] Florian, M., Chen, Y. *A coordinate descent method for bilevel O-D matrix estimation problems.* Technical Report CRT-807, Centre de Recherche sur les Transports, 1993.

[28] Fortuny-Amat, J., McCarl, B. "A representation and economic interpretation of a two-level programming problem". *Journal of the Operational Research Society*, v. 32, pp. 783–792, 1981.

[29] Hansen, P., Jaumard, B., Savard, G. "New branch-and-bound rules for linear bilevel programming". *SIAM Journal on Scientific and Statistical Computing*, v. 13, pp. 1194–1217, 1992.

[30] Herskovits, J., Leontiev, A. "An interior point algorithm for convex bilevel programming problems". In *Annales 1er Encuentro Latino Iberoamericano de Optimización*, pp. 389–390, Concépcion, Chile, 1997.

[31] Herskovits, J., Leontiev, A., Santos, G. "A mathematical programming algorithm for optimal design of elastic solids in contact". In *Anals 2nd World Congress of Structural and Multidisciplinary optimization*, Zakopane, Polônia, 1997.

[32] Hobbs, B., Nelson, S. "A nonlinear bilevel model for analysis of electric utility demand-side planning issues". *Annals of Operations Research*, v. 34, pp. 255–274, 1992.

[33] Horst, R., Pardalos, P., Thoai, N. *Introduction to global optimization.* John Wiley & Sons, Inc., Holanda, 1995.

[34] Horst, R., Tuy, H. *Global optimization: deterministic approaches.* Spring-Verlag, Berlin, 1990.

[35] Islam, S. "Sustainable economic development in the australian energy sector: findings of the australian energy planning system optimization model (aepsom)". *Renewable and Sustainable Energy Reviews*, v. 1, pp. 229–238, 1997.

[36] Jeroslow, R. "The polynomial hierarchy and a simple model for competitive analysis". *Mathematical Programming*, v. 32, pp. 146–164, 1985.

[37] Júdice, J., Faustino, A. "A sequential LCP method for bilevel linear programming". *Annals of Operations Research*, v. 34, pp. 89–106, 1992.

[38] Júdice, J., Mitra, G. "Reformulation of mathematical programming problems as linear complementarity problems and investigation of their solution methods". *Journal of Optmization: Theory and Applications*, v. 57, n. 1, pp. 123–149, 1988.

[39] Kirsch, U. "Two-level optimization of prestressed structures". *Engineering Structures*, v. 19, n. 4, pp. 309–317, 1997.

[40] Konno, H. "A cutting plane algorithm for solving bilinear programs". *Mathematical Programming*, v. 11, pp. 14–27, 1976.

[41] Li, G., Zhou, R., Duan, L., Chen, W. "Multiobjective and multilevel optimization for steel frames". *Engineering Structures*, v. 21, pp. 519–529, 1999.

[42] Migdalas, A. "Bilevel programming in traffic planning: models, methods and challenge". *Journal of Global Optimization*, v. 7, pp. 381–405, 1995.

[43] Mitten, L. "Branch-and-bound methods: General formulation and properties". *Journal of Operational Research*, v. 18, pp. 315–344, 1970.

[44] Önal, H. "A modified simplex approach for solving bilevel linear programming problems". *European Journal of Operational Research*, v. 67, pp. 126–135, 1993.

[45] Önal, H., Darmawan, D., Johnson, S. "A multilevel analysis of agricultural credit distribution in East Java, Indonesia". *Computers and Operations Research*, v. 22, pp. 227–236, 1995.

[46] Rockafellar, R. *Convex Analysis.* Princeton University Press, Nova Jersey, 2a. ed., 1972.

[47] Sherali, H., Shetty, C. "A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts". *Mathematical Programming*, v. 19, pp. 14–31, 1980.

[48] Shimizu, K., Ishizuka, Y., Bard, J. *Nondifferentiable and two-level mathematical programming.* Kluwer Academic Publishers, 1997.

[49] Stackelberg, H. *The theory of the market economy.* Oxford University Press, New York, Oxford, 1952.

[50] Taha, H. *Integer Programming: Teory, Applications, and Computations.* Academic Press, 1975.

[51] Tuy, H. "Concave programming under linear constraints". *Soviet Mathematics*, v. 5, pp. 1437–1440, 1964.

[52] Tuy, H. "On nonconvex optimization problems with separated nonconvex variables". *Journal of Global Optimization*, v. 2, pp. 133–144, 1992.

[53] Tuy, H., Ghannadan, S. "A new branch-and-bound method for bilevel linear programs". In Migdalas, A., Pardalos, P., Värbrand, P. (eds), *Multilevel optimization: algorithms and applications.* Nonconvex Optimization and its Applications 20: 231–249, Kluwer Academic Publishers, 1998.

[54] Vaish, H., Shetty, C. "The bilinear programming problem". *Naval Research Logistics Quarterly*, v. 23, pp. 303–309, 1976.

[55] Vicente, L., Calamai, P. "Bilevel and multilevel programming: a bibliography review". *Journal of Global Optimization*, v. 5, pp. 291–306, 1994.

[56] White, D., Anandalingam, G. "A penalty function approach for solving bi-level linear programs". *Journal of Global Optimization*, v. 3, pp. 397–419, 1993.

[57] Wolf, D., Smeers, Y. "Optimal dimensioning of pipe networks with application to gas transmission metworks". *Operations Research*, v. 44, n. 4, pp. 596–608, 1996.

[58] Yang, H., Yagar, S. "Traffic assignment and signal control in saturated road networks". *Transportation Research*, v. 29A, n. 2, pp. 125–139, 1995.

[59] Zhang, J., Zhu, D. "A bilevel programming method for pipe network optimization". *SIAM Journal on Optimization*, v. 6, n. 3, pp. 838–857, 1996.