Submetido para TEMA

Um novo algoritmo para soluções ótimas locais do problema linear de dois níveis

Autor

Resumo. Neste artigo, apresentamos um algoritmo para encontrar soluções ótimas locais dos problemas lineares de dois níveis. A cada ponto viável corrente, o método busca por melhores soluções no conjunto dos pontos que se encontram em suas faces adjacentes. Em cada passo tenta-se encontrar as faces adjacentes de maior dimensão, na esperança de acelerar o processo. Uma prova de corretude do método é fornecida, e testes computacionais foram realizados.

Palavras-chave. Programação em dois níveis, problema linear de dois níveis, solução ótima local.

1. Introdução

Problemas de dois níveis foram considerados primeiramente por Bracken and McGill ([4]). São compostos por dois agentes que tomam decisões em níveis de hierarquia distintos. Enquanto o agente no topo da hierarquia (líder) toma sua decisão, o agente subordinado (escravo ou seguidor) procura, dentre as opções que lhe são oferecidas pelo líder, aquelas que melhor lhe convém. Neste caso, o escravo pode tomar decisões que cooperam com os interesses do líder (problema otimista) ou decisões que vão de encontro aos interesses do líder (problema pessimista). Assim, problemas envolvendo relações de hierarquia entre dois níveis de decisão podem ser modelados como problemas de dois níveis. Há casos, por exemplo, em economia, projeto de redes, engenharia e teoria dos jogos [4].

Mais especificamente, formulamos o problema de dois níveis como

$$\min_{x,y} \{F(x,y)\,;\, G(x,y) \leq 0,\ y \in \arg\min_{y} \{f(x,y)\,;\, g(x,y) \leq 0\}\}.$$

O objetivo do líder é portanto minimizar $F(x,y)$, enquanto que, fixado x, o escravo procura minimizar $f(x,y)$. Neste artigo, consideramos o caso particular onde as funções F, f, G e g são lineares: o *problema linear de dois níveis*, dado por

$$\text{PLDN:} \quad \min_{x,y} \{c_1^t x + d_1^t y\} \tag{1.1}$$

$$\text{s.a. } A_1 x + B_1 y \leq b_1, \quad x \geq 0 \tag{1.2}$$

$$y \in \arg\min_{y} d_2^t y \tag{1.3}$$

$$\text{s.a. } A_2 x + B_2 y \leq b_2, \quad y \geq 0 \tag{1.4}$$

onde $c_1 \in \mathbb{R}^{n_x}$, $d_1, d_2 \in \mathbb{R}^{n_y}$, $A_1 \in \mathbb{R}^{m_u \times n_x}$, $B_1 \in \mathbb{R}^{m_u \times n_y}$, $A_2 \in \mathbb{R}^{m_l \times n_x}$, $B_2 \in \mathbb{R}^{m_l \times n_y}$, $b_1 \in \mathbb{R}^{m_u}$ e $b_2 \in \mathbb{R}^{m_l}$. Hansen et al. em 1992 mostraram que PLDN é NP-difícil (veja [6, 7]).

Algoritmos para PLDN foram propostos. Dentre eles, o método do k-ésimo melhor vértice de Bialas e Karwan, os algoritmos *branch and bound* de Bard e Moore, e posteriormente de Hansen et al., o método sequencial de Júdice e Faustino baseado em problemas de complementariedade linear, métodos de penalização de Önal, de White e Anandalingam, e de Campêlo, o algoritmo de Tuy et al., métodos de pontos interiores, e métodos baseados em programação multi-objetivo. Para uma revisão desses métodos, recomendamos [1, 3, 5, 6, 9]. Também, metaheurísticas foram utilizadas [2, 10, 11, 13]. Para uma revisão bibliográfica das aplicações e métodos de solução, recomendamos [4, 8].

Neste artigo, propomos um novo método para encontrar soluções ótimas locais de PLDN. A cada ponto viável corrente, o método busca por melhores soluções no conjunto dos pontos que se encontram em suas faces adjacentes. Em cada passo tenta-se encontrar as faces adjacentes de maior dimensão, na esperança de acelerar o processo. Para isso, escrevemos o problema do nível inferior em suas condições de Karush-Kuhn-Tucker (KKT). Nesse sentido, o algoritmo assemelha-se com o método de conjuntos ativos em programação não linear, com a diferença que aqui as condições KKT são restrições de um outro problema (do nível superior).

Este trabalho é organizado como segue. Na seção 2, apresentamos brevemente os conceitos e resultados que embasam nosso método. Na seção 3, o algoritmo e a prova de sua corretude, e na seção 4, testes computacionais. Finalmente, nossas conclusões são expostas na seção 5.

2. Definições e propriedades

Consideremos o PLDN. Chamamos (1.1), (1.2) de *problema do nível superior* e (1.3), (1.4) de *problema do nível inferior*. Nesse sentido, (1.1) e (1.3) são as funções objetivo dos níveis superior e inferior, respectivamente, e (1.2) e (1.4) são as restrições dos níveis superior e inferior, respectivamente. Consideramos também os seguintes conjuntos:

- os conjuntos viáveis dos problemas do nível superior, $\Omega_u = \{(x, y); A_1x + B_1y \leq b_1, x \geq 0\}$, e do nível inferior, $\Omega_l = \{(x, y); A_2x + B_2y \leq b_2, y \geq 0\}$;
- $\operatorname{graf} S = \{(x, y); y \in S(x)\}$, onde $S(x) = \arg\min_y \{d_2^t y; (x, y) \in \Omega_l\}$.

Dizemos que (x, y) é *racional* se $(x, y) \in \operatorname{graf} S$ e *admissível* se $(x, y) \in \Upsilon = \operatorname{graf} S \cap \Omega_u$ (*conjunto admissível*). Nesse estágio, PLDN pode ser reescrito como

$$\min_{x,y} c_1^t x + d_1^t y \quad \text{s.a.} \quad (x, y) \in \Upsilon. \tag{2.1}$$

Um importante conceito para o desenvolvimento de nosso algoritmo é o de face de um conjunto poliedral:

Definição 2.1 ([12]). *Dado um conjunto poliedral* $D = \{z \in \mathbb{R}^n; Az \leq b\}$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, *um subconjunto* $Q \subset D$ *é dito ser uma* face *de* D *se existir um conjunto* $J \subset \{1, \ldots, m\}$ *tal que*

$$Q = Q(J) = \{z \in D; A_j z = b_j, j \in J\}.$$

Dados J *e sua face associada* $Q(J)$, *dizemos que* Q *é* face d-dimensional *de* D *(ou* face de dimensão d*),* $0 \leq d \leq n$, *se existir* $z_0 \in Q(J)$ *tal que* $A_j z_0 < b_j, \forall j \notin J$, *e a matriz* $[A]_{i\in J}$, *cujas linhas são as linhas de* A *que correspondem aos índices em* J, *tem posto* $n - d$. *Mais ainda, dizemos que* $Q(J)$ *é* face não degenerada *se* $|J| = n - d$.

Comumente diz-se que uma face 0-dimensional é um vértice, e uma face 1-dimensional é uma *aresta*. Diremos também que uma face Q é *adjacente* à z se $z \in Q$. Notemos ainda que cada face é um conjunto poliedral.

A fim de resolver PLDN localmente, reescrevemos o problema do nível inferior em suas condições de otimalidade de Karush-Kuhn-Tucker. Isto é, $y \in S(x)$ se, e somente se

$$\begin{aligned} &A_2x + B_2y \leq b_2, \quad y \geq 0, \quad \lambda \geq 0, \quad \lambda^t(A_2x + B_2y - b_2) = 0, \\ &B_2^t\lambda + d_2 \geq 0, \quad y^t(B_2^t\lambda + d_2) = 0. \end{aligned} \tag{2.2}$$

Para conjuntos $I \subset \{1, \ldots, n_y\}$ e $J \subset \{1, \ldots, m_l\}$, consideremos o conjunto $M(I, J)$ das soluções $(x, y, \lambda) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}^{m_l}$ do sistema

$$(A_2x + B_2y - b_2)_i = 0, i \in J \tag{2.3}$$
$$(A_2x + B_2y - b_2)_i \leq 0, i \notin J \tag{2.4}$$
$$y_j = 0, j \in I, \quad y_j \geq 0, j \notin I \tag{2.5}$$
$$\lambda_i = 0, i \notin J, \quad \lambda_i \geq 0, i \in J \tag{2.6}$$
$$(B_2^t\lambda + d_2)_j = 0, j \notin I, \quad (B_2^t\lambda + d_2)_j \geq 0, j \in I. \tag{2.7}$$

Agora, o conjunto das soluções de (2.2) é a união dos finitos $M(I, J)$ [6]. Com isso, estabelecemos o conhecido resultado:

Teorema 2.1 ([6]). $\operatorname{graf} S$ *é igual à uma união finita de faces de* Ω_l.

Assim, diremos que uma face é *racional* se for face de $\operatorname{graf} S$, isto é, se só contiver pontos racionais. Considerando (2.1), nosso métodos buscará, dentre as faces racionais, soluções ótimas locais que pertençam à Ω_u.

3. O algoritmo

Aqui, estamos supondo que todas as faces de Ω são não degeneradas. Procurar então faces de Ω_l adjacentes a um ponto e que estão em $\operatorname{graf} S$ pode ser feito obrigando $\lambda_i = 0$ para certo $i \notin J$ no sistema (2.3)-(2.7). Ao fazer isso, liberamos uma restrição para que seja não ativa. No algoritmo seguinte, utilizamos três conjuntos

para controle dos índices das restrições do nível inferior: L para índices de restrições que foram liberadas em todas as faces adjacentes estudadas num ponto corrente; Q para índices de restrições que foram liberadas no estudo de uma face adjacente a um ponto corrente; e R para índices de restrições que foram testadas (quanto à racionalidade da face que induzem) no estudo de uma face adjacente a um ponto corrente.

A seguir, apresentamos o algoritmo, e logo após a prova de sua corretude. Em seguida, aplicamo-lo a um problema exemplo.

Algoritmo 1

Entrada: PLDN viável. Supõe-se todas as faces de Ω não degeneradas.
Saída: Uma solução ótima local de PLDN ou a constatação de sua ilimitabilidade.

---

1. (Inicialização) Calcule $(x^0, y^0) \in \Upsilon$ usando uma heurística. Faça $L \leftarrow \emptyset$, $k \leftarrow 0$ e defina

$$\Lambda^k = \{i; (A_2x^k + B_2y^k - b_2)_i = 0\} \cup \{m_l + i; y_i^k = 0\}.$$

   Se (x^0, y^0) for vértice de Ω_l, vá para o passo 2. Caso contrário, execute o passo 5 com $Q = \emptyset$.

2. Faça $Q \leftarrow \emptyset$ e $R \leftarrow \emptyset$. Vá para o passo 3.

3. (Detecção de faces racionais adjacentes) Se $Q \neq \emptyset$, escolha $j \in \Lambda^k \backslash R$.

   Se $Q = \emptyset$, escolha se possível $j \in \Lambda^k \backslash (L \cup R)$. Se tal escolha não for possível, vá para o passo 4.

   Verifique se o sistema

$$\begin{aligned} (B_2^t\lambda + d_2)_i &= 0, \qquad m_l + i \in \{j\} \cup \complement\Lambda \\ (B_2^t\lambda + d_2)_i &\geq 0, \qquad m_l + i \in \Lambda \backslash \{j\} \\ \lambda_i &= 0, \qquad i \in \{j\} \cup \complement\Lambda \\ \lambda_i &\geq 0, \qquad i \in \Lambda \backslash \{j\} \end{aligned}$$

   admite solução, onde $\Lambda = \Lambda^k \backslash R$ e $\complement\Lambda = \{1, \ldots, m_l + n_y\} \backslash \Lambda$. Se admite solução, digamos λ^k, faça

$$R \leftarrow R \cup A, \quad Q \leftarrow Q \cup A \quad \text{e} \quad L \leftarrow L \cup A$$

   onde $A = \{i \in \Lambda^k; \lambda_i^k = 0\} \cup \{m_l + i \in \Lambda^k; (B_2^t\lambda^k + d_2)_i = 0\} \subset \Lambda^k$. Se o sistema anterior não admite solução, faça $R \leftarrow R \cup \{j\}$. Se $\Lambda^k \backslash R = \emptyset$ vá para o passo 4. Se $\Lambda^k \backslash R \neq \emptyset$ repita o passo 3.

4. (Teste de parada) Se $Q \neq \emptyset$, vá para o passo 5. Se $Q = \emptyset$ pare e retorne (x^k, y^k) como solução ótima local de PLDN.

5. (Passo de minimização) Resolva

$$\begin{aligned}
&\min_{x,y} \{c_1^t x + d_1^t y\} \\
&\text{s.a. } A_1 x + B_1 y \leq b_1, \quad x \geq 0 \\
&\qquad (A_2 x + B_2 y - b_2)_i = 0, \quad i \in \Lambda \\
&\qquad (A_2 x + B_2 y - b_2)_i \leq 0, \quad i \notin \Lambda \\
&\qquad y_i = 0, \quad m_l + i \in \Lambda \\
&\qquad y_i \geq 0, \quad m_l + i \notin \Lambda
\end{aligned}$$

onde $\Lambda = \Lambda^k \backslash Q$. Observe que o problema é viável pois (x^k, y^k) é um ponto viável seu. Se o problema é ilimitado, pare e reporte que PLDN é ilimitado. Caso contrário, tome uma solução ótima (x^{k+1}, y^{k+1}) sua. Se $c_1^t x^{k+1} + d_1^t y^{k+1} < c_1^t x^k + d_1^t y^k$, faça $k \leftarrow k+1$, $L \leftarrow \emptyset$ e vá para o passo 2. Senão, vá para o passo 2.

Teorema 3.1. *Suponha que PLDN seja viável, com Ω sem faces degeneradas. Então o Algoritmo 1 converge em um número finito de passos a uma solução ótima local, ou conclui que PLDN é ilimitado.*

Demonstração. Tomemos $(x^0, y^0) \in \Upsilon$. O algoritmo gera um número finito $(x^0, y^0), \ldots, (x^r, y^r)$ de pontos em Υ visto que gerado (x^{k+1}, y^{k+1}) com $c_1^t x^{k+1} + d_1^t y^{k+1} < c_1^t x^k + d_1^t y^k$, o algoritmo não volta a analisar $(x^0, y^0), \ldots, (x^k, y^k)$; para todo (x^k, y^k), o laço 3 termina pois a cada iteração, R é acrescido até que $\Lambda^k \backslash R = \emptyset$; para todo (x^k, y^k) no passo 5, ou paramos com a constatação da ilimitabilidade de PLDN, ou passamos a um novo $(x^{k+1}, y^{k+1}) \neq (x^k, y^k)$ ou repetimos esse processo com sucessivos fracassos de diminuição da função $c_1^t x + d_1^t y$, e neste caso L é acrescido até que $\Lambda^k \backslash (L \cup R) = \emptyset$. No último caso, de acordo com o passo 3, $Q = \emptyset$ e o algoritmo termina com o passo 4. Esta situação sempre ocorre pois existem finitas faces em Ω.

Se no passo 5 o problema é ilimitado, PLDN também o é, e a conclusão é correta. Por outro lado, suponha que não se conclue a ilimitabilidade de PLDN, e seja (x^r, y^r) um ponto obtido no fim do algoritmo. Seja U^r o conjunto dos índices das restrições do nível superior ativas em (x^r, y^r). Tomemos $\delta > 0$ tal que para cada $(x, y) \in B_\delta(x^r, y^r) = \{(\tilde{x}, \tilde{y}); \|(\tilde{x}, \tilde{y}) - (x^r, y^r)\| < \delta\}$ tenhamos

$$\begin{aligned}
&(A_1 x + B_1 y - b_1)_i < 0, \quad x_{i-m_u} > 0, \quad \forall i \notin U^r \\
&(A_2 x + B_2 y - b_2)_i < 0, \quad y_{i-m_l} > 0, \quad \forall i \notin \Lambda^r.
\end{aligned} \tag{3.1}$$

Para cada $j \in \Lambda^r$, consideremos a face u_j de Ω_u, adjacente à (x^r, y^r), definida

por

$$\begin{aligned} u_j = \{(x,y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y}; &(A_1x + B_1y - b_1)_i = 0, \quad i \in U^r \backslash \{j\}, \\ &(A_1x + B_1y - b_1)_i \leq 0, \quad i \notin U^r \backslash \{j\}, \\ &x_i = 0, \quad m_u + i \in U^r \backslash \{j\}, \\ &x_i \geq 0, \quad m_u + i \notin U^r \backslash \{j\}\}. \end{aligned}$$

Seja u^r a face de Ω_u cujas restrições/variáveis de igualdade têm índices em U^r. Analogamente, para cada $j \in \Lambda^r$, consideremos a face q_j de Ω_l, adjacente a (x^r, y^r),

$$\begin{aligned} q_j = \{(x,y) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y}; &(A_2x + B_2y - b_2)_i = 0, \quad i \in \Lambda^r \backslash \{j\}, \\ &(A_2x + B_2y - b_2)_i \leq 0, \quad i \notin \Lambda^r \backslash \{j\}, \\ &y_i = 0, \quad m_l + i \in \Lambda^r \backslash \{j\}, \\ &y_i \geq 0, \quad m_l + i \notin \Lambda^r \backslash \{j\}\}, \end{aligned}$$

e a face q^r de Ω_l cujas restrições/variáveis de igualdade têm índices em Λ^r.

Agora, suponhamos por contradição que (x^r, y^r) não seja solução ótima local de PLDN. Existe então um $(\overline{x}, \overline{y}) \in B_\delta(x^r, y^r) \cap \Upsilon$ com $c_1^t\overline{x} + d_1^t\overline{y} < c_1^t x^r + d_1^t y^r$. Afirmamos que existem $n_x + n_y - |U^r| - |\Lambda^r|$ vetores linearmente independentes (LI), que denotaremos por a_j, $j \in R$, tais que $(x^r, y^r) + a_j \in q^r \cap u^r$. De fato, consideremos a matriz

$$C = \begin{bmatrix} \begin{bmatrix} A_1 & B_1 \\ -I_{n_x} & 0 \end{bmatrix}_{j \in U^r} \\ \begin{bmatrix} A_2 & B_2 \\ 0 & -I_{n_y} \end{bmatrix}_{j \in \Lambda^r} \end{bmatrix}$$

e a aplicação linear T associada a C. Pela não degeneração das faces de Ω, C tem posto igual ao seu número de linhas, a saber, $|U^r| + |\Lambda^r|$. Assim, $\dim \mathrm{Im}(T) = |U^r| + |\Lambda^r|$ e o Teorema do Núcleo e da Imagem nos diz que $\dim \mathrm{Ker}(T) = n_x + n_y - |U^r| - |\Lambda^r|$. Logo, existem $n_x + n_y - |U^r| - |\Lambda^r|$ vetores d_j LI satisfazendo $Cz = 0$. Como

$$C \begin{bmatrix} x^r \\ y^r \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} b_1 \\ 0 \end{bmatrix}_{j \in U^r} \\ \begin{bmatrix} b_2 \\ 0 \end{bmatrix}_{j \in \Lambda^r} \end{bmatrix} = b',$$

e levando em consideração (3.1), existem $\mu_j > 0$, $j \in R$, suficientemente pequenos tais que $(x^r, y^r) + \mu_j d_j \in q^r \cap u^r$. Tomemos então $a_j = \mu_j d_j$.

Utilizando, se necessário, o processo de ortogonalização de Gram-Schmidt, podemos supor sem perda de generalidade que os vetores a_j são ortogonais entre si. Observamos ainda que todos a_j são ortogonais às linhas de C, visto que são soluções do sistema homogêneo $Cz = 0$. Então os vetores $C_1, \ldots, C_{|U^r|+|\Lambda^r|}$ e a_j, $j \in R$, são LI. De fato, se fosse $a_j = \sum_{i \neq j} \eta_i a_i + \sum_i \beta_i C_i$ teríamos

$$0 < a_j^t a_j = \sum_{i \neq j} \eta_i (a_i^t a_j) + \sum_i \beta_i (C_i a_j) = 0.$$

Da mesma forma, se $C_j = \sum_i \eta_i a_i + \sum_{i\neq j} \beta_i C_i$, pelo menos um η_k é não nulo pois as linhas de C são LI. Daí

$$0 = C_j a_k = \sum_i \eta_i (a_i^t a_k) + \sum_{i\neq j} \beta_i (C_i a_k) = \eta_k (a_k^t a_k) \neq 0.$$

Com isto, para cada $j \in R$, construímos uma restrição de desigualdade σ_j ativa em (x^r, y^r) e que é satisfeita por $(\overline{x}, \overline{y})$, como segue. Se $a_j^t \begin{bmatrix} \overline{x}^t & \overline{y}^t \end{bmatrix}^t \leq a_j^t \begin{bmatrix} x^{rt} & y^{rt} \end{bmatrix}^t$ tomamos $\sigma_j : a_j^t \begin{bmatrix} x^t & y^t \end{bmatrix}^t \leq a_j^t \begin{bmatrix} x^{rt} & y^{rt} \end{bmatrix}^t$ e caso contrário, tomamos $\sigma_j : (-a_j^t) \begin{bmatrix} x^t & y^t \end{bmatrix}^t \leq (-a_j^t) \begin{bmatrix} x^{rt} & y^{rt} \end{bmatrix}^t$. Consideremos então, para cada $j \in R$, os conjuntos

$$w_j = \{(x,y); (x,y) \text{ satisfaz } \sigma_j\}, \quad \overline{w}_j = \{(x,y); \sigma_j \text{ é ativa em } (x,y)\}.$$

Tomemos $W = \bigcap_{i\in R} w_i$. Assim, (x^r, y^r) é vértice não degenerado de $\Omega^+ = \Omega_u \cap \Omega_l \cap W = \Omega \cap W$. Definindo $W_j = w_j \cap \bigcap_{i\in R\setminus\{j\}} \overline{w}_i$ e $\overline{W} = \bigcap_{i\in R} \overline{w}_i$, as $n_x + n_y$ arestas de Ω^+ emanando de (x^r, y^r) são $q_j \cap u^r \cap \overline{W}$, $j \in \Lambda^r$, $q^r \cap u_j \cap \overline{W}$, $j \in U^r$, e $q^r \cap u^r \cap W_j$, $j \in R$. Também,

$$\begin{array}{l} \emptyset \neq q_j \cap u^r \cap \overline{W} \subset q_j \cap \Omega_u, \quad \forall j \in \Lambda^r, \\ \emptyset \neq q^r \cap u_j \cap \overline{W} \subset q^r \cap \Omega_u, \quad \forall j \in U^r, \quad \text{e} \\ \emptyset \neq q^r \cap u^r \cap W_j \subset q^r \cap \Omega_u, \quad \forall j \in R. \end{array} \tag{3.2}$$

Para simplificar a escrita, vamos supor que os conjuntos de índices U^r, Λ^r e R sejam disjuntos, e que $U^r \cap \Lambda^r \cap R = \{1, \ldots, n_x + n_y\}$. As arestas de Ω^+ emanando de (x^r, y^r) definem direções $\gamma_1, \ldots, \gamma_{n_x+n_y}$. Sendo $\gamma = (\overline{x}, \overline{y}) - (x^r, y^r)$, existem $\alpha_j \geq 0$ tais que

$$\gamma = \sum_i \alpha_i \gamma_i. \tag{3.3}$$

Como $(\overline{x}, \overline{y}) \in B_\delta(x^r, y^r)$, segue de (3.1) que todas as restrições não ativas em (x^r, y^r) também são não ativas em $(\overline{x}, \overline{y})$. Por outro lado, pode ocorrer de restrições ativas em (x^r, y^r) serem não ativas em $(\overline{x}, \overline{y})$. Sejam portanto $\overline{U} \subset U^r$ e $\overline{\Lambda} \subset \Lambda^r$ os conjuntos dos índices das restrições dos níveis superior e inferior, respectivamente, que são ativas em (x^r, y^r) mas não em $(\overline{x}, \overline{y})$. Para cada j, escrevendo $\gamma_j = (x_j, y_j) - (x^r, y^r)$, de (3.1) e do fato de (x^r, y^r) ser vértice não degenerado de Ω^+, podemos supor que (x_j, y_j) realiza como desigualdade estrita todas as restrições de

desigualdade em sua respectiva aresta. Temos então, para $j \in \Lambda^r \backslash \overline{\Lambda}$, se $j \leq m_l$,

$$\begin{aligned}
0 &= (A_2\overline{x} + B_2\overline{y} - b_2)_j \\
&= A_2\left(x^r + \sum_i \alpha_i(x_i - x^r)\right)_j + B_2\left(y^r + \sum_i \alpha_i(y_i - y^r)\right)_j - (b_2)_j \\
&= \underbrace{(A_2x^r + B_2y^r - b_2)_j}_{0} + \sum_i \alpha_i \left[A_2(x_i - x^r) + B_2(y_i - y^r)\right]_j \\
&= \sum_i \alpha_i (A_2x_i + B_2y_i - b_2)_j = \alpha_j \underbrace{(A_2x_j + B_2y_j - b_2)_j}_{<0} \Rightarrow \alpha_j = 0,
\end{aligned}$$

e se $j > m_l$,

$$0 = \overline{y}_{j-m_l} = \underbrace{(y^r)_{j-m_l}}_{0} + \sum_i \alpha_i (y_i - y^r)_{j-m_l} = \alpha_j \underbrace{(y_j)_{j-m_l}}_{>0} \Rightarrow \alpha_j = 0.$$

Ou seja, $\alpha_j = 0$ sempre que $j \in \Lambda^r \backslash \overline{\Lambda}$. Logo, de (3.3) obtemos

$$\gamma = \sum_{i \in \overline{\Lambda}} \alpha_i \gamma_i + \sum_{i \in U^r \cup R} \alpha_i \gamma_i. \tag{3.4}$$

Agora, como $(\overline{x}, \overline{y})$ e (x^r, y^r) são admissíveis para PLDN, as faces q^r e q_j, $j \in \overline{\Lambda}$, são racionais. Segue portanto de (3.2) que as arestas de Ω^+ emanando de (x^r, y^r) são formadas de pontos admissíveis de PLDN, exceto possivelmente as relativas aos índices em $\Lambda^r \backslash \overline{\Lambda}$. Logo as direções γ_j, $j \in U^r \cup \overline{\Lambda} \cup R$, são viáveis para PLDN.

Ao fim do algoritmo, o passo 5 garante que (x^r, y^r) é solução ótima do problema

$$\min_{x,y} \{c_1^t x + d_1^t y\} \quad \text{s.a.} \quad (x, y) \in q^r \cap \Omega_u. \tag{3.5}$$

É importante observarmos que se $r = 0$, a execução do passo 5 logo após o cálculo de (x^0, y^0) garantiria tal afirmação. Não é necessário executar o passo 5 se (x^0, y^0) for vértice de Ω_l pois teríamos $q^0 = \{(x^0, y^0)\}$. Com isto, as duas últimas inclusões em (3.2) dizem que as direções γ_j, $j \in U^r \cup R$, não são de descida. Também, ao fim do algoritmo temos $\Lambda^r \backslash (L \cup R) = Q = \emptyset$. Isto significa que cada índice em Λ^r foi analisado pelo algoritmo, mesmo que vários índices foram escolhidos de uma só vez no passo 3 (neste caso todas as faces q_j com esses índices são analisadas de uma só vez). Neste caso o passo 5 garante que (x^r, y^r) é solução ótima dos problemas

$$\min_{x,y} \{c_1^t x + d_1^t y\} \quad \text{s.a.} \quad (x, y) \in q_j \cap \Omega_u,$$

para todos $j \in \overline{\Lambda}$ ($j \in \overline{\Lambda}$ é escolhido com sucesso no passo 3 pois q_j é racional). Daí segue da primeira inclusão em (3.2) que as direções γ_j, $j \in \overline{\Lambda}$, não são de descida.

Finalmente mostremos que (x^r, y^r) é solução ótima local de PLDN. Pela nossa suposição inicial, a direção $\gamma = (\overline{x}, \overline{y}) - (x^r, y^r)$ é de descida para a função $c_1^t x + d_1^t y$.

Segue de (3.4) que

$$0 > \left[\begin{array}{cc} c_1^t & d_1^t \end{array}\right] \gamma = \sum_{i \in \overline{\Lambda}} \alpha_i \left[\begin{array}{cc} c_1^t & d_1^t \end{array}\right] \gamma_i + \sum_{i \in U^r \cup R} \alpha_i \left[\begin{array}{cc} c_1^t & d_1^t \end{array}\right] \gamma_i \geq 0.$$

Concluímos portanto que (x^r, y^r) é solução ótima local de PLDN. □

Observação 1: No passo 1, é necessário um ponto admissível inicial. É sabido que encontrar um tal ponto é um problema NP-difícil no caso das restrições do nível superior dependerem das variáveis do nível inferior [12]. No caso contrário, isso pode ser feito resolvendo no máximo dois problemas de programação linear: o primeiro, a relaxação obtida quando desconsideramos a função objetivo do nível inferior; e, caso tenhamos obtido (x^*, y^*) não admissível, um segundo problema, constituído pelo problema do nível inferior quando se fixa $x = x^*$ [12].

Observação 2: Podemos adaptar o Algoritmo 1 a problemas que não possuem restrições de não negatividade $y \geq 0$. Neste caso, as condições de otimalidade para o problema do nível inferior exigem que as restrições $B_2^t \lambda + d_2 \geq 0$ sejam de igualdade. Podemos então desconsiderar a parcela $\{m_l + i; y_i = 0\}$ na definição de Λ^k e reescrever os passos 3 e 5 de modo conveniente.

4. Testes computacionais

Nessa seção apresentamos alguns testes computacionais. Os problemas foram gerados do seguinte modo: as entradas de c_1, d_1, d_2, B_2 e A_2 foram geradas aleatoriamente em $[-10, 10]$. Cada entrada i de b_2 é calculada por $q\sqrt{|c_i|}$, onde c_i é a soma dos termos da linha i da matriz $[A_2\ B_2]$ e q é um número aleatório em $[1, 10]$. Os problemas não possuem restrições do nível superior, isto é, A_1, B_1 e b_1 são nulas. São consideradas ainda restrições de não negatividade $x, y \geq 0$. Vale ressaltar que Still [12] gera seus problemas de forma semelhante.

A Tabela 1 resume as características dos problemas testados. As colunas n_x, n_y, m_l são como anteriormente. Também,

- $|\Lambda^0|$ é a quantidade de restrições ativas no primeiro ponto admissível (x^0, y^0);
- F_0 é o valor da função objetivo do nível superior no término do passo 1;
- F é o valor da função objetivo do nível superior no fim do algoritmo;
- LB é o limitante inferior obtido da relaxação em que se desconsidera a função objetivo (1.3) do nível inferior;
- t é o tempo de processamento em segundos;
- P3 é a quantidade das iterações globais em que foram escolhidos mais de um índice no passo 3;

n_x	n_y	m_l	$\|\Lambda^0\|$	F_0	F	LB	t	P3	Gl
4	2	6	6	-17,26	-17,26	-17,26	$<$ 0,01	0	3
4	6	10	9	-52,52	-153,51	-385,51	$<$ 0,01	0	3
4	10	14	13	-2314,65	-2314,65	-2314,65	$<$ 0,01	5	6
6	4	10	9	78,51	-549,51	-809,17	$<$ 0,01	4	5
6	10	16	12	-228,33	-294,41	-299,45	0,01	4	5
8	4	12	12	-24,95	-24,95	-24,95	$<$ 0,01	2	3
8	10	18	15	-166,64	-378,62	-482,93	$<$ 0,01	3	4
16	16	32	24	-116,83	-516,86	-753,45	0,02	10	11
20	20	40	33	-778,08	-988,71	-1001,84	0,04	12	13
20	40	100	51	-428,10	-821,58	-943,81	0,21	15	16
60	40	200	66	-1527,18	-1527,18	-1527,18	1,28	18	19
60	100	200	129	-4146,71	-5645,12	-5691,23	12,18	60	61
100	70	250	119	-4094,77	-4094,79	-4094,79	9,39	33	34
100	150	350	202	-5231,95	-7098,34	-7389,40	88,52	86	87

Tabela 1: Testes computacionais

- Gl é o número de iterações globais.

Os testes foram realizados em um computador AMD X2 5600+ (2,8 Ghz por núcleo), com 2Gb de memória, no sistema GNU/Linux. Para a resolução de problemas de programação linear e do sistema do passo 3, foi utilizado o CPLEX 12.3.

Em 5 dos 14 problemas, o ponto admissível inicial, calculado seguindo a Observação 1, atingiu LB (e portanto é ótimo). Em todos os outros 9 problemas, o algoritmo melhorou significativamente a solução. Os resultados indicam que o algoritmo se comporta melhor em problemas com n_y grande em relação à n_x, e com muitas restrições do nível inferior, onde observamos *gaps* entre F e LB menores. Cabe observar que na maioria dos problemas, em quase a totalidade das iterações globais, foram escolhidos mais de um índice no passo 3. Isso significa que no passo 5, a busca por melhores soluções é feita sobre faces racionais de grande dimensão. É esperado portanto que o algoritmo explore uma grande quantidade de soluções por iteração. Por isso acreditamos que o métodos é promissor.

5. Conclusões

Nesse artigo foi proposto um novo método para encontrar soluções ótimas locais do problema linear de dois níveis. O método baseia-se na reformulação do problema do nível inferior em suas condições KKT. Assim, faces adjacentes ao ponto corrente são analisadas de modo a obter uma que seja racional e que tenha maior dimensão possível. Com isso, espera-se que o método analise mais soluções por iteração, e consiga soluções ótimas locais melhores. Testes indicam que faces com grandes dimensões são encontradas pelo algoritmo em praticamente todas as iterações. Acreditamos

portanto que o método é promissor, quando comparado a outros como o de George Still [12], que analisa apenas faces com 1 dimensão a mais que a corrente.

Estudos podem ser feitos no sentido de escolher os índices no passo 3 de uma melhor maneira. Acreditamos que se conseguirmos antever quais faces são mais promissoras, poderíamos obter melhores soluções ótimas locais. Trabalhos futuros poderão seguir esta linha.

Referências

[1] J. F. Bard. *Practical bilevel optimization: algorithms and applications*, volume 30 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers, 1998.

[2] H. I. Calvete, C. Galé, and P. M. Mateo. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 188(1):14–28, 2008.

[3] M. B. C. Campêlo. *Programação linear em dois níveis: uma abordagem teórica e computacional*. PhD thesis, COPPE/UFRJ, Rio de Janeiro, December 1999. In portuguese.

[4] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operation Research*, 153(1):235–256, 2007.

[5] C. H. M. de Sabóia, M. B. C. Campêlo, and S. Scheimberg. A computational study of global algorithms for bilevel linear programming. *Numerical Algorithms*, 35(2-4):155–173, 2004.

[6] S. Dempe. *Foundations of bilevel programming*, volume 61 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers, 2002.

[7] X. Deng. Complexity issues in bilevel linear programming. In A. Migdalas, P. M. Pardalos, and P. Värbrand, editors, *Multilevel optimization: algorithms and applications*, Nonconvex Optimization and Its Applications, chapter 6, pages 149–164. Kluwer Academic Publishers, 1998.

[8] C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1):3–38, 2009.

[9] J. Glackin, J. G. Ecker, and M. Kupferschmid. Solving bilevel linear programs using multiple objective linear programming. *Journal of Optimization Theory And Application*, 140(2):197–212, 2009.

[10] R. J. Kuoa and C. C. Huang. Application of particle swarm optimization algorithm for solving bi-level linear programming problem. *Computers and Mathematics with Applications*, 58(4):678–685, 2009.

[11] K. H. Sabin and A. R. Ciric. A dual temperature simulated annealing approach for solving bilevel programming problems. *Computers and Chemical Engineering*, 23(1):11–25, 1998.

[12] G. Still. Linear bilevel problems: genericity results and an efficient method for computing local minima. *Mathematical Methods of Operations Research*, 55(3):383–400, 2002.

[13] U. P. Wen and A. D. Huang. A simple tabu search method to solve the mixed-integer linear bilevel programming problem. *European Journal of Operational Research*, 88(3):563–571, 1996.