

Um algoritmo de *branch-and-bound* baseado em programação linear para programação não linear inteira mista convexa

Wendel Melo

Faculdade de Computação
Universidade Federal de Uberlândia
wendelmelo@ufu.br

Marcia Fampa

Instituto de Matemática e COPPE
Universidade Federal do Rio de Janeiro
fampa@cos.ufrj.br

Fernanda Raupp

Laboratório Nacional de Computação Científica
fernanda@lncc.br

RESUMO

Neste trabalho, introduzimos um novo algoritmo para Programação Não Linear Inteira Mista convexa fundamentado em dois algoritmos bem sucedidos da literatura: Plano de Corte Estendido (PCE) e Branch-And-Bound baseado em Programação Linear e Não Linear (BB-PL/PNL). Desse modo, a abordagem proposta reúne as marcantes características de ser um método totalmente de primeira ordem, como PCE, com o avançado esquema de aproximação linear usado por BB-PL/PNL. Resultados computacionais apontaram a efetividade do novo método sobre um conjunto de 343 instâncias de teste, cujo desempenho foi bastante competitivo com os dos algoritmos mencionados.

PALAVRAS CHAVE: Programação Não Linear Inteira Mista, Plano de Corte Estendido, Branch-And-Bound.

Área de classificação principal: Otimização Combinatória.

ABSTRACT

In this work, we introduce a new algorithm for convex mixed integer nonlinear programming based on two successful algorithms from the literature: Extended Cutting Plane (ECP) and Linear and Nonlinear Programming based Branch-And-Bound (LP/NLP-BB). The approach proposed joins the characteristics of being a first-order method, such as ECP, with the advanced linear approximation scheme used by LP/NLP-BB. Computational results show the effectiveness of the new method on a set of 343 test instances, whose performance was competitive with the two other algorithms mentioned.

KEYWORDS: Mixed Integer Nonlinear Programming, Extended Cutting Plane, Branch-And-Bound.

Main area: Combinatorial Optimization.

1 Introdução

Neste trabalho, abordamos o seguinte problema de Programação Não Linear Inteira Mista (PNLIM) convexa:

$$\begin{aligned} (P) \quad & \min_{x, y} f(x, y) \\ \text{s. a} \quad & g(x, y) \leq 0, \\ & x \in X, y \in Y \cap \mathbb{Z}^{n_y}, \end{aligned} \tag{1}$$

onde X e Y são subconjuntos poliédricos de $\mathbb{R}^{n_x}$ e $\mathbb{R}^{n_y}$, respectivamente, Y é limitado, $f : \mathbb{R}^{n_x+n_y} \rightarrow \mathbb{R}$ e $g : \mathbb{R}^{n_x+n_y} \rightarrow \mathbb{R}^m$ são funções convexas e continuamente diferenciáveis. Aqui, assumimos que o problema (P) possui solução ótima.

Diversas aplicações envolvendo modelos baseados em PNLIM têm aparecido na literatura [Leyffer et al., 2009]. A dificuldade envolvida no tratamento do problema (P) , bem como sua grande aplicabilidade em situações variadas justificam a pesquisa por algoritmos eficientes para a sua resolução. Nesse contexto, diversas abordagens têm sido propostas para solucionar problemas de PNLIM [Bonami et al., 2008; Borchers and Mitchell, 1994; Duran and Grossmann, 1986; Fletcher and Leyffer, 1994; Geoffrion, 1972; Kronqvist et al., 2016; Leyffer, 2001; Melo et al., 2014; Quesada and Grossmann, 1992; Westerlund and Pettersson, 1995]. Dentre essas diferentes abordagens, os algoritmos da classe de aproximação linear merecem especial destaque. Esta classe de algoritmos resolve um problema de PNLIM convexo aproximando-o por uma sequência de problemas de Programação Linear Inteira Mista (PLIM), cujas resoluções fornecem limitantes inferiores (no caso de minimização) para o problema considerado. Tais aproximações são obtidas por meio de derivadas de primeira ordem e se baseiam na convexidade das funções presentes em (P) .

Os algoritmos de aproximação linear possuem a grande vantagem de se valerem diretamente de todo o avanço e maturidade já alcançados na área de PLIM através do uso de sofisticados pacotes computacionais como, por exemplo, **Cplex**, **Gurobi** e **Xpress**. Todavia, em muitos casos, como, por exemplo, [Duran and Grossmann, 1986; Fletcher and Leyffer, 1994; Geoffrion, 1972; Kronqvist et al., 2016; Quesada and Grossmann, 1992], esses algoritmos exigem a resolução de uma série de problemas de Programação Não Linear contínua (PNL), cuja resolução pode demandar alta carga de esforços e está sujeita a toda uma gama de possíveis dificuldades numéricas. Ademais, os procedimentos computacionais utilizados para estas resoluções de problemas de PNL podem falhar em encontrar uma solução ótima (ou mesmo uma solução viável, em alguns casos) para o problema abordado, mesmo em casos onde tal solução ótima exista, além de, possivelmente, requererem o cálculo de derivadas de segunda ordem das funções do problema, o que pode ser demasiadamente custoso em muitas situações práticas. Nesse contexto, o algoritmo de Plano de Corte Estendido, ou PCE, ([Melo et al., 2017; Westerlund and Pettersson, 1995]) merece destaque por não resolver nenhum problema de PNL ao longo de sua execução, evitando assim todos os possíveis problemas associados com tal atividade e se constituindo então em um método totalmente de primeira ordem. A versão de PCE apresentada por [Melo et al., 2017] apresentou bons resultados computacionais comparada à versão original de PCE de [Westerlund and Pettersson, 1995] e ao algoritmo Aproximação Externa (AE) de [Duran and Grossmann, 1986] e [Fletcher and Leyffer, 1994]. Contudo, no estudo conduzido por [Melo et al., 2018], o algoritmo Branch-And-Bound baseado em Programação Linear e Não Linear (BB-PL/PNL) de [Quesada and Grossmann, 1992] apresentou os melhores resultados dentre os diversos algoritmos considerados pelo estudo no ambiente de testes adotado. Embora BB-PL/PNL resolva problemas de PNL ao longo de sua execução, o mesmo possui um esquema bem sofisticado para aproximar (P) linearmente em um esquema de BB, o que tem se mostrado crucial para seu desempenho superior aos dos demais algoritmos em muitos casos.

Nesse trabalho, apresentamos um novo algoritmo para Programação Não Linear Mista convexa que busca utilizar o sofisticado esquema de aproximação linear de BB-PL/PNL se baseando nas estratégias de evolução de PCE. Desse modo, construímos um algoritmo de BB baseado apenas em Programação Linear, que apresenta a principal vantagem de PCE (ser um método totalmente de primeira ordem) aliada à principal vantagem de BB-PL/PNL (um esquema rebuscado para a aproximação linear de (P)). Denominamos nossa abordagem como *Branch-And-Bound baseado em Programação Linear*, ou, BB-PL.

Este texto está organizado da seguinte forma. A Seção 2 apresenta o algoritmo PCE. Em seguida é apresentado o algoritmo AE na Seção 3, que é a base do algoritmo BB-PL/PNL, que é apresentado em sequência na Seção 4. Nosso algoritmo BB-PL é então apresentado na Seção 5. Por fim, resultados computacionais e conclusões são apresentados na Seção 6.

2 Plano de Corte Estendido

O algoritmo de Plano de Corte Estendido foi originalmente proposto por [Westerlund and Pettersson, 1995] e se baseia na aproximação do problema (P) por um problema de Programação Linear Inteira Mista (PLIM), denominado como problema *mestre*. Para facilitar o entendimento do problema mestre, observamos que (P) pode ser reformulado como um problema com função objetivo linear através da introdução de uma variável auxiliar α :

$$\begin{aligned} (\bar{P}) \quad & \min_{\alpha, x, y} \quad \alpha \\ \text{s. a} \quad & f(x, y) \leq \alpha \\ & g(x, y) \leq 0 \\ & x \in X, y \in Y \cap \mathbb{Z}^{n_y}. \end{aligned} \quad (2)$$

Uma vez que as restrições de $(\bar{P})$ são convexas, ao linearizarmos as mesmas por meio da série de Taylor sobre um dado ponto qualquer $(\bar{x}, \bar{y}) \in X \times Y$, obtemos as seguintes desigualdades válidas para $(\bar{P})$:

$$\nabla f(\bar{x}, \bar{y})^T \begin{pmatrix} x - \bar{x} \\ y - \bar{y} \end{pmatrix} + f(\bar{x}, \bar{y}) \leq \alpha, \quad (3)$$

$$\nabla g(\bar{x}, \bar{y})^T \begin{pmatrix} x - \bar{x} \\ y - \bar{y} \end{pmatrix} + g(\bar{x}, \bar{y}) \leq 0. \quad (4)$$

Note que o ponto $(\bar{x}, \bar{y})$ não precisa necessariamente satisfazer as restrições de (P) . Repare ainda que em $(\bar{x}, \bar{y})$, as linearizações aproximam de forma exata as funções originais de (P) .

De posse destas informações, obtemos o problema mestre a partir de um conjunto $L = \{(x^1, y^1), \dots, (x^k, y^k)\}$ com k pontos de linearização que se constitui em uma relaxação do problema original (P) :

$$\begin{aligned} (M^L) \quad & \min_{\alpha, x, y} \quad \alpha \\ \text{s. a} \quad & \nabla f(x^j, y^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} + f(x^j, y^j) \leq \alpha, \quad \forall (x^j, y^j) \in L \\ & \nabla g(x^j, y^j)^T \begin{pmatrix} x - x^j \\ y - y^j \end{pmatrix} + g(x^j, y^j) \leq 0, \quad \forall (x^j, y^j) \in L \\ & x \in X, y \in Y \cap \mathbb{Z}^{n_y}. \end{aligned} \quad (5)$$

Seja $(\hat{\alpha}, \hat{x}, \hat{y})$ uma das soluções ótimas de (M^L) . Apontamos que o valor $\hat{\alpha}$ é um limite inferior válido para $(\bar{P})$ e (P) . Se $(\hat{\alpha}, \hat{x}, \hat{y})$ for viável para $(\bar{P})$, temos que o valor $\hat{\alpha}$ também

será um limite superior para $(\bar{P})$ e (P) . Nesse caso, uma vez que $(\hat{\alpha}, \hat{x}, \hat{y})$ provê o mesmo valor como limite inferior e superior para $(\bar{P})$, esta solução será ótima para $(\bar{P})$. Consequentemente, $(\hat{x}, \hat{y})$ será uma solução ótima para (P) . Por outro lado, se $(\hat{\alpha}, \hat{x}, \hat{y})$ não for viável para $(\bar{P})$, é necessário acrescentar desigualdades válidas em (M^L) de modo a cortar essa solução de sua região viável, fortalecendo assim a relaxação fornecida por esse problema. Para alcançar este objetivo, o algoritmo PCE adota a estratégia de acrescentar a própria solução $(\hat{x}, \hat{y})$ ao conjunto L .

Uma nova versão de PCE foi apresentada por [Melo et al., 2017]. Nessa nova versão, apresentada como Algoritmo 1, são acrescentados ao conjunto L pontos de linearização adicionais obtidos através da fixação das variáveis inteiras do problema (M^L) e de novas resoluções deste (linhas 13 e 17), que, nesta situação, se torna um problema de programação linear, mantendo assim o método como sendo puramente de primeira ordem.

Entrada:	(P) : problema de PNLIM, ϵ_c : tolerância de convergência.
Saída:	(x^*, y^*) : solução ótima de (P) .
1	$z^l = -\infty$;
2	$z^u = +\infty$;
3	Escolha um ponto de linearização inicial (x^0, y^0) ;
4	$L = \{(x^0, y^0)\}$;
5	$k = 1$;
6	Seja (M^L) o problema mestre construído a partir de (P) sobre os pontos em L ;
7	Seja $(M_{\hat{y}^k}^L)$ o problema obtido ao fixar a variável y de (M^L) em $\hat{y}^k$;
8	enquanto $z^u - z^l > \epsilon_c$ faça
9	Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de (M^L) ;
10	$z^l = \hat{\alpha}^k$;
11	$L = L \cup (\hat{x}^k, \hat{y}^k)$;
12	se $(M_{\hat{y}^k}^L)$ <i>é viável</i> então
13	Seja (α^k, x^k) uma solução ótima de $(M_{\hat{y}^k}^L)$;
14	se $(x^k, \hat{y}^k)$ <i>é viável para</i> (P) e $f(x^k, \hat{y}^k) < z^u$ então
15	$(x^*, y^*) = (x^k, \hat{y}^k)$;
16	$z^u = f(x^k, \hat{y}^k)$;
17	$L = L \cup (x^k, \hat{y}^k)$;
18	$k = k + 1$;

Algoritmo 1: Algoritmo de Plano de Corte Estendido (versão de [Melo et al., 2017]).

Apesar das alterações introduzidas no Algoritmo 1, ainda assim, o mesmo manteve a característica de PCE de não garantir convergência em número finito de iterações. No entanto, os resultados em [Melo et al., 2017] apontaram desempenho bastante superior desta nova versão de PCE em comparação com a original no ambiente de testes adotado.

3 Aproximação Externa

O algoritmo Aproximação Externa (AE) foi proposto originalmente por [Duran and Grossmann, 1986] e posteriormente aperfeiçoado por [Fletcher and Leyffer, 1994]. Assim como PCE, o fundamento principal de AE consiste em adotar o problema mestre (M^L) , e, a cada iteração, acrescentar um novo ponto de linearização em L até que o limite inferior fornecido por (M^L) se torne suficientemente próximo do melhor limite superior conhecido para (P) . Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de (M^L) na iteração k . Uma tentativa de obtenção de um limite superior para (P) consiste na resolução do problema $(P_{\hat{y}^k})$, que é o problema de

PNL obtido a partir de (P) fixando-se y no valor inteiro $\hat{y}^k$:

$$\begin{aligned} (P_{\hat{y}^k}) \quad & \min_x \quad f(x, \hat{y}^k) \\ \text{s. a} \quad & g(x, \hat{y}^k) \leq 0 \\ & x \in X. \end{aligned} \quad (6)$$

Se $(P_{\hat{y}^k})$ for viável, seja $(\tilde{x}^k, \hat{y}^k)$ uma de suas soluções ótimas. Nesse caso, $f(\tilde{x}^k, \hat{y}^k)$ é um limite superior válido para (P) e $(\bar{P})$, e o ponto $(\tilde{x}^k, \hat{y}^k)$ é adicionado ao conjunto L . Observe que $(\tilde{x}^k, \hat{y}^k)$ é a melhor solução para (P) e $(\bar{P})$ (ou uma das melhores, caso $(P_{\hat{y}^k})$ tenha múltiplas soluções ótimas) tendo $\hat{y}^k$ como valor para y . Caso $(P_{\hat{y}^k})$ seja inviável, então o seguinte problema de viabilidade é resolvido:

$$\begin{aligned} (P_{\hat{y}^k}^F) \quad & \min_{u, x} \quad \sum_{i=1}^m u_i \\ \text{s. a} \quad & g(x, \hat{y}^k) \leq u \\ & u \geq 0, \\ & x \in X, u \in \mathbb{R}^m. \end{aligned} \quad (7)$$

Seja $(\tilde{u}^k, \tilde{x}^k)$ uma solução ótima de $(P_{\hat{y}^k}^F)$ no contexto descrito. Então, o ponto $(\tilde{x}^k, \hat{y}^k)$ é adicionado ao conjunto L . Após a atualização de L , com a inclusão de $(\tilde{x}^k, \hat{y}^k)$ se o problema $(P_{\hat{y}^k})$ for viável, ou com a inclusão de $(\tilde{x}^k, \hat{y}^k)$ caso contrário, o algoritmo inicia uma nova iteração, tendo como critério de parada uma tolerância máxima para a diferença entre os melhores limites inferior e superior obtidos. Conforme demonstrado em [Bonami et al., 2008], assumindo que as condições de KKT são satisfeitas nas soluções de $(P_{\hat{y}^k})$ e $(P_{\hat{y}^k}^F)$, essa estratégia de atualização do conjunto L faz com que, a cada iteração, a solução ótima do problema mestre $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ seja cortada pelas novas linearizações se a mesma não for solução viável para $(\bar{P})$. Foi demonstrado ainda que essa estratégia garante que uma determinada solução $\hat{y}^k$ para a variável inteira y não seja visitada mais que uma vez pelo algoritmo, exceto se a mesma fizer parte da solução ótima de (P) (nesse caso, a solução poderia ser visitada no máximo duas vezes). Como o número de soluções inteiras é finito por hipótese, já que Y é limitado, o algoritmo garantidamente encontra uma solução ótima de (P) em um número finito de iterações. Assim, em comparação com o algoritmo ECP, AE tende a despendar um número menor de iterações, com o custo de necessitar resolver, a cada iteração, um ou dois problemas de PNL. De modo geral, em muitos casos práticos, AE tende a possuir melhor desempenho que PCE devido ao número reduzido de iterações.

O Algoritmo 2 traz o algoritmo AE. Na linha 3, um ponto de linearização inicial precisa ser obtido. Uma boa escolha é utilizar a solução da relaxação contínua de (P) , que é o problema de PNL obtido a partir de (P) ao se relaxar sua restrição de integralidade de y . Note que esta mesma estratégia poderia ser adotada no algoritmo PCE (Algoritmo 1). Entretanto, se isso for feito, PCE passará a necessitar de uma rotina de PNL e possivelmente de informação proveniente de derivada de segunda ordem.

4 Branch-And-Bound baseado em PL e PNL

Em [Quesada and Grossmann, 1992] é apresentado um algoritmo de aproximação linear baseado na resolução de problemas de Programação Linear (PL) e Programação Não Linear (PNL) em um esquema de *Branch-and-Bound* (*Branch-and-Bound* baseado em PL/PNL, ou BB-PL/PNL). A estratégia de atualização do conjunto L é a mesma de AE, isto é, fundamentada na resolução dos problemas $(P_{\hat{y}^k})$ e $(P_{\hat{y}^k}^F)$. A principal motivação deste algoritmo vem da observação de que, em geral, o problema mestre (M^L) é resolvido sucessivas vezes

Entrada: (P) : problema de PNLIM, ϵ_c : tolerância de convergência.

Saída: (x^*, y^*) : solução ótima de (P) .

```

1   $z^l = -\infty$ ;
2   $z^u = +\infty$ ;
3  Escolha um ponto de linearização inicial  $(x^0, y^0)$ , (usualmente a solução ótima da relaxação contínua
   de  $(P)$ );
4   $L = \{(x^0, y^0)\}$ ;
5   $k = 1$ ;
6  Seja  $(M^L)$  o problema mestre construído a partir de  $(P)$  sobre os pontos em  $L$ ;
7  Seja  $(P_{\hat{y}^k})$  o problema PNL obtido ao fixar a variável  $y$  de  $(P)$  em  $\hat{y}^k$ ;
8  Seja  $(P_{\hat{y}^k}^F)$  o problema PNL de viabilidade obtido a partir de  $(P_{\hat{y}^k})$ ;
9  enquanto  $z^u - z^l > \epsilon_c$  faça
10     Seja  $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$  uma solução ótima de  $(M^L)$ ;
11      $z^l = \hat{\alpha}^k$ ;
12     se  $(P_{\hat{y}^k})$  é viável então
13         Seja  $x^k$  uma solução ótima de  $(P_{\hat{y}^k})$ ;
14         se  $f(x^k, y^k) < z^u$  então
15              $z^u = f(x^k, y^k)$ ;
16              $(x^*, y^*) = (x^k, y^k)$ ;
17     senão
18         Seja  $x^k$  uma solução ótima de  $(P_{\hat{y}^k}^F)$ ;
19      $L = L \cup (x^k, \hat{y}^k)$ ;
20      $k = k + 1$ ;

```

Algoritmo 2: Algoritmo de Aproximação Externa (AE).

nos algoritmos de aproximação linear. Essa resolução é comumente realizada por meio de um procedimento caixa-preta especializado, geralmente uma sub-rotina que implementa alguma metodologia baseada em *Branch-and-Bound* (BB). Estas sub-rotinas podem vir de avançados pacotes de PLIM como **Cplex**, **Gurobi** ou **Xpress**. Quesada e Grossmann propuseram então abrir este procedimento caixa-preta, integrando a resolução do problema mestre com a resolução de (P) através de um esquema único de BB. Nesse esquema de BB, em cada partição Y^k explorada, o algoritmo resolve a seguinte relaxação contínua do problema mestre:

$$\begin{aligned}
 & \left(\tilde{M}_{Y^k}^L \right) \min_{\alpha, x, y} \alpha \\
 & \text{s. a} \quad \nabla f(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + f(x^k, y^k) \leq \alpha, \quad \forall (x^k, y^k) \in L \\
 & \quad \nabla g(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + g(x^k, y^k) \leq 0, \quad \forall (x^k, y^k) \in L \\
 & \quad x \in X, y \in Y^k.
 \end{aligned} \tag{8}$$

Observamos que o problema mestre é dinamicamente atualizado através da inclusão de novos pontos de linearização em L , que se originam da resolução dos problemas $(P_{\hat{y}^k})$ e $(P_{\hat{y}^k}^F)$ quando uma solução inteira por ventura é encontrada.

O algoritmo *Branch-and-Bound* baseado em PL/PNL é mostrado como Algoritmo 3. Note que apenas podas por inviabilidade e limite são realizadas. Quando uma solução inteira é obtida na resolução de $(\tilde{M}_{Y^k}^L)$ (linha 13), o algoritmo entra em um laço de atualização e resolução do problema mestre até que uma solução não inteira seja obtida (linhas 14-24). Apontamos que a atualização do problema mestre não segue a ideia de particionamento do espaço do algoritmo BB, isto é, quando linearizações são acrescentadas na exploração de uma determinada partição do espaço, as novas restrições passarão a ser impostas a todos

```

Entrada:  $(P)$ : Problema de PNLIM,  $\epsilon_c$ : tolerância de convergência.
Saída:  $(x^*, y^*)$ : solução ótima para  $(P)$ .
1  $z^u = +\infty$ ;
2 Escolha um ponto de linearização inicial  $(x^0, y^0)$ , (usualmente a solução ótima de  $(\tilde{P})$  (relaxação
   contínua de  $(P)$ ));
3  $L = \{(x^0, y^0)\}$ ;
4  $Y^0 = Y$ ;
5 Seja  $N = \{0\}$  a lista inicial de nós em aberto da enumeração de BB;
6 Seja  $l_b^i$  o limitante inferior do nó  $i$ ;
7  $l_b^0 = -\infty$ ;
8  $i = 0$ ;
9 enquanto  $N \neq \emptyset$  faça
10   Selecione um nó  $k$  de  $N$ ;
11    $N = N \setminus \{k\}$ ;
12   se  $(\tilde{M}_{Y^k}^L)$  é viável então
13     Seja  $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$  uma solução ótima de  $(\tilde{M}_{Y^k}^L)$ ;
14     enquanto  $\hat{y}^k$  possui valor inteiro e  $\hat{\alpha}^k < z^u - \epsilon_c$  faça
15       se  $(P_{\hat{y}^k})$  é viável então
16         Seja  $\tilde{x}^k$  uma solução ótima de  $(P_{\hat{y}^k})$ ;
17         se  $f(\tilde{x}^k, \hat{y}^k) < z^u$  então
18            $z^u = f(\tilde{x}^k, \hat{y}^k)$ ;
19            $(x^*, y^*) = (\tilde{x}^k, \hat{y}^k)$ ;
20            $N = N \setminus \{j : l_b^j \geq z^u - \epsilon_c\}$ ; // Poda por limite
21         senão
22           Seja  $(\tilde{u}^k, \tilde{x}^k)$  uma solução ótima de  $(P_{\hat{y}^k}^F)$ ;
23            $L = L \cup \{(\tilde{x}^k, \hat{y}^k)\}$ ;
24           Seja  $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$  uma solução ótima de  $(\tilde{M}_{Y^k}^L)$ ;
25         se  $\hat{\alpha}^k < z^u - \epsilon_c$  então
26           // Ramificação
27           Selecione uma variável  $y_j$  com valor não inteiro  $\hat{y}_j^k$ ;
28            $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \leq \lfloor \hat{y}_j^k \rfloor\}$ ;
29            $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \geq \lceil \hat{y}_j^k \rceil\}$ ;
30            $l_b^{i+1} = l_b^{i+2} = \hat{\alpha}^k$ ;
31            $N = N \cup \{i+1, i+2\}$ ;
32            $i = i+2$ ;

```

Algoritmo 3: *Branch-and-Bound* baseado em PL/PNL.

os subproblemas $(\tilde{M}_{Y^k}^L)$ resolvidos daquela iteração em diante, independentemente do ramo em que estejam na árvore de BB.

À primeira vista, pode parecer difícil conseguir implementar o Algoritmo 3 de forma tão eficiente quanto AE, pois, ao abrir o procedimento de resolução do problema mestre, o algoritmo BB-PL/PNL perde, em princípio, a habilidade de utilizar as engenhosas estratégias que avançados *solvers* de PLIM empregam para ampliar sua eficiência. No entanto, estes mesmos *solvers* fornecem um meio prático de implementar algoritmos como este através do uso de *lazy constraints* juntamente com funções *callback* que permitem modificar o comportamento padrão de seu procedimento de BB. Adicionando as novas linearizações do problema mestre como *lazy constraints*, é possível implementar o Algoritmo 3 facilmente utilizando pacotes como **Cplex** ou **Gurobi**, continuando assim a se valer de todos os recursos disponíveis nesses *softwares*. Desse modo, uma vez que considera um único procedimento de BB, o *branch-and-bound* para PLIM não precisa ser reiniciado a cada iteração, conforme ocorre em AE. Assim, espera-se que o Algoritmo 3 apresente uma carga de computação reduzida em comparação com este último. De fato, resultados apresentados em [Melo et al.,

2018] apontaram melhor desempenho deste algoritmo em relação aos demais algoritmos de PNLIM em geral nas instâncias de teste consideradas.

5 Nosso algoritmo de Branch-And-Bound baseado em PL

Nesta seção, apresentamos um novo algoritmo para PNLIM convexa que integra o avançado esquema de aproximação linear de BB-PL/PNL (Algoritmo 3) com a estratégia de atualização do conjunto de pontos de linearização de PCE (Algoritmo 1). Dessa forma, obtemos um método totalmente de primeira ordem, fundamentado em um *branch-and-bound* baseado em aproximação linear e que não requer a resolução de problemas de PNL, evitando assim possíveis dificuldades atreladas ao tratamento deste tipo de problema. Denominamos nossa abordagem como *Branch-and-Bound* baseado em PL, ou BB-PL, e a descrevemos em detalhes no Algoritmo 4.

Entrada: (P): Problema de PNLIM, ϵ_c: tolerância de convergência. Saída: (x^*, y^*): solução ótima para (P). <pre> 1 $z^u = +\infty$; 2 Escolha um ponto de linearização inicial (x^0, y^0); 3 $L = \{(x^0, y^0)\}$; 4 $Y^0 = Y$; 5 Seja $N = \{0\}$ a lista inicial de nós em aberto da enumeração de BB; 6 Seja l_b^i o limitante inferior do nó i; 7 $l_b^0 = -\infty$; 8 $i = 0$; 9 enquanto $N \neq \emptyset$ faça 10 Seleccione um nó k de N; 11 $N = N \setminus \{k\}$; 12 se $(\tilde{M}_{Y^k}^L)$ <i>é viável</i> então 13 Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $(\tilde{M}_{Y^k}^L)$; 14 enquanto $(\tilde{M}_{Y^k}^L)$ <i>é viável</i> e $\hat{y}^k$ <i>possui valor inteiro</i> e $\hat{\alpha}^k < z^u - \epsilon_c$ faça 15 se $(\hat{x}^k, \hat{y}^k)$ <i>é viável para</i> (P) e $f(\hat{x}^k, \hat{y}^k) < z^u$ então 16 $z^u = f(\hat{x}^k, \hat{y}^k)$; 17 $(x^*, y^*) = (\hat{x}^k, \hat{y}^k)$; 18 $N = N \setminus \{j : l_b^j \geq z^u - \epsilon_c\}$; // Poda por limite 19 $L = L \cup \{(\hat{x}^k, \hat{y}^k)\}$; 20 Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de $(\tilde{M}_{Y^k}^L)$; 21 se $\hat{\alpha}^k < z^u - \epsilon_c$ então 22 // Ramificação 23 Seleccione uma variável y_j com valor não inteiro $\hat{y}_j^k$; 24 $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \leq \lfloor \hat{y}_j^k \rfloor\}$; 25 $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j \geq \lceil \hat{y}_j^k \rceil\}$; 26 $l_b^{i+1} = l_b^{i+2} = \hat{\alpha}^k$; 27 $N = N \cup \{i+1, i+2\}$; 28 $i = i + 2$; </pre>	
--	--

Algoritmo 4: Nosso algoritmo *Branch-and-Bound* baseado em PL (BB-PL).

Como pode ser conferido nos Algoritmos 3 e 4, a principal diferença entre BB-PL/PNL e BB-PL é que este último considera a resolução do próprio problema $(\tilde{M}_{Y^k}^L)$ no lugar dos problemas $(P_{\hat{y}^k})$ e $(P_{\hat{y}^k}^F)$. Pode ser facilmente aferido que, se a solução $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ obtida para $(\tilde{M}_{Y^k}^L)$ na linha 20 do Algoritmo 4 for viável para (P) com $\hat{y}^k$ inteiro e $\hat{\alpha}^k = f(\hat{x}^k, \hat{y}^k)$, então os cortes gerados em BB-PL pela introdução de $(\hat{x}^k, \hat{y}^k)$ em L (linha 19) serão semelhantes aos de BB-PL/PNL, já que, nesse caso, $\hat{x}^k$ também será solução ótima para $(P_{\hat{y}^k})$.

Desse modo, temos que os cortes gerados por nosso algoritmo BB-PL não são mais fortes que os gerados por BB-PL/PNL. Todavia, esperamos que essa aparente desvantagem seja compensada pelo fato de BB-PL não resolver problemas de PNL, dispensando também assim o possível cálculo de derivadas de segunda ordem. A convergência de BB-PL pode ser verificada de modo intuitivo a partir da própria convergência de PCE, e do esquema de *branch-and-bound* utilizado, que não descarta nenhuma partição do espaço indevidamente. Por fim, pode-se perceber ainda que BB-PL também pode ser implementado por meio de *lazy constraints* e *callbacks* de usuário em pacotes PLIM avançados de forma mais simples que BB-PL/PNL, já que problemas de PNL não são resolvidos.

6 Resultados Computacionais e Conclusões

Foram obtidos resultados computacionais a partir da aplicação de PCE, AE, BB-PL/PNL e nossa abordagem BB-PL sobre um conjunto de 343 instâncias de teste de problemas de PNLIM convexa oriundas dos seguintes repositórios [CMU-IBM, 2012; Leyffer, 2013; World, 2014]. A Tabela 1 traz informações estatísticas sobre os problemas de teste. É notável o alto percentual de restrições lineares presentes nas instâncias de modo geral.

Tabela 1: Estatísticas sobre os problemas teste.

	min	max	média	mediana
variáveis	2	107222	975,34	114
variáveis inteiras (%)	0,01%	100%	52%	40%
restrições	0	108217	1184,65	211
restrições lineares (%)	0,00%	100%	89%	95%

Os algoritmos mencionados foram implementados em C++ 2011 e compilados com o compilador ICPC 16.0.0. Para a resolução de problemas de PLIM, adotamos o solver Cplex 12.6.0. Para a resolução de problemas de PNL, adotamos o solver Mosek 8.0.0.79. Os testes rodaram em um computador com processador core i7 4790 (3,6 GHz) e 16 GB de RAM sobre o sistema operacional Open Suse Linux 13.1. Todos os algoritmos foram configurados para serem executados por uma única *thread* de processamento, o que implica que os mesmos foram executados por um único processador a cada instante de tempo na máquina utilizada para os testes. O tempo de processamento (CPU) de cada algoritmo em cada instância de teste foi limitado a 4 horas. Foram adotados os valores 10^{-6} e 10^{-3} como tolerâncias de convergência absoluta e relativa, respectivamente, para todos os algoritmos. Como pontos de linearização iniciais para os algoritmos PCE e BB-LP, foram tomadas as origens do sistema cartesiano $((0)^{n_x+n_y})$. Para variáveis com restrições de caixa nos problemas, foi tomado como solução inicial seu respectivo limite inferior.

A Figura 1 traz uma comparação relativa entre os algoritmos com respeito ao tempo de processamento (CPU) dispendido no conjunto de instâncias de teste considerados. Note que os dados estão normalizados em relação ao melhor resultado obtido dentre todas as abordagens para cada instância. No eixo horizontal, a abscissa indica o número de vezes que o resultado obtido pelo algoritmo foi maior que o melhor resultado dentre os algoritmos com relação ao tempo computacional. No eixo vertical, a ordenada indica o percentual de instâncias alcançadas por cada abordagem. Mais especificamente, se a curva de um determinado algoritmo passa pelo ponto (α, τ) , isto indica que para $\tau\%$ das instâncias, o resultado obtido pelo algoritmo em questão é menor ou igual a α vezes o melhor tempo computacional dentre todos os algoritmos.

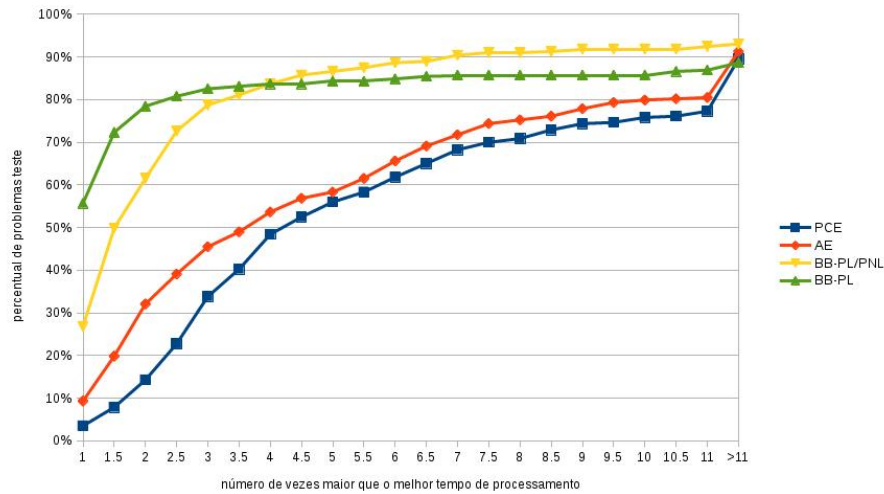


Figura 1: Comparação relativa de tempo de processamento (CPU) para os algoritmos.

Note, por exemplo, que a curva de AE passa pelo ponto (1, 10%). Isto significa que, para 10% das instâncias de teste consideradas, AE conseguiu o melhor resultado com relação ao tempo computacional (uma vez maior que o melhor resultado). A seguir, a curva passa pelo ponto (1,5, 20%), indicando que AE conseguiu resolver 20% das instâncias gastando até 50% a mais de tempo que o melhor algoritmo em cada instância (1,5 vezes maior que o melhor resultado). Desta forma, a grosso modo, podemos afirmar que um algoritmo se saiu melhor que os demais no quesito em questão quanto mais sua curva está “acima” das curvas dos demais algoritmos no gráfico.

Através da Figura 1, é possível notar que nossa abordagem BB-PL foi a mais eficiente para a maior parte das instâncias consideradas, resolvendo 56% das mesmas com o menor tempo de processamento dentre todas as abordagens. BB-PL/PNL, por sua vez, apresentou o menor tempo de processamento para 27% das instâncias. Com relação a AE e PCE, esse número foi de 9% e 3%, respectivamente. É válido notar que nem todas as instâncias de teste puderam ser totalmente resolvidas por cada abordagem no tempo máximo estipulado. BB-PL foi capaz de solucionar por completo cerca de 89% das instâncias, ficando levemente atrás das demais abordagens. BB-PL/PNL resolveu cerca de 93% das instâncias, enquanto AE e PCE resolveram cerca de 91% e 90% das instâncias, respectivamente. É possível perceber ainda que BB-PL apresenta o melhor desempenho quando se considera tempo de processamento até 4 vezes maior que o melhor resultado. A partir daí, ele é superado por BB-PL/PNL. BB-PL foi também capaz de resolver 78% das instâncias de teste gastando até duas vezes mais que o melhor tempo de processamento, ao passo que BB-PL/PNL, AE e PCE resolveram, respectivamente, 62%, 32% e 14% dos problemas nessa mesma proporção de tempo. Por fim, é válido notar que o desempenho de AE domina o de PCE nesse conjunto de instâncias de teste. A principal explicação decorre do fato de que AE gera cortes mais fortes para o problema mestre com a resolução de $(P_{\hat{y}^k}^F)$ e $(P_{\hat{y}^k}^F)$. Entretanto, no contexto dos experimentos aqui descritos, nossa abordagem BB-PL conseguiu obter um considerável ganho de desempenho em relação à PCE ao adotar sua estratégia de cortes no avançado esquema de aproximação linear de BB-PL/PNL. Sob certos aspectos, é possível afirmar ainda que BB-PL também alcançou ganhos em relação a BB-PL/PNL nos testes aqui realizados.

A seguir, avaliamos os desempenhos no subconjunto de problemas teste com maiores percentuais de restrições não lineares. Separamos então todas as 92 instâncias do grupo de

343 com mais de 20% de restrições não lineares para gerar um novo gráfico comparativo entre as abordagens com relação ao tempo de processamento, exibido na Figura 2.

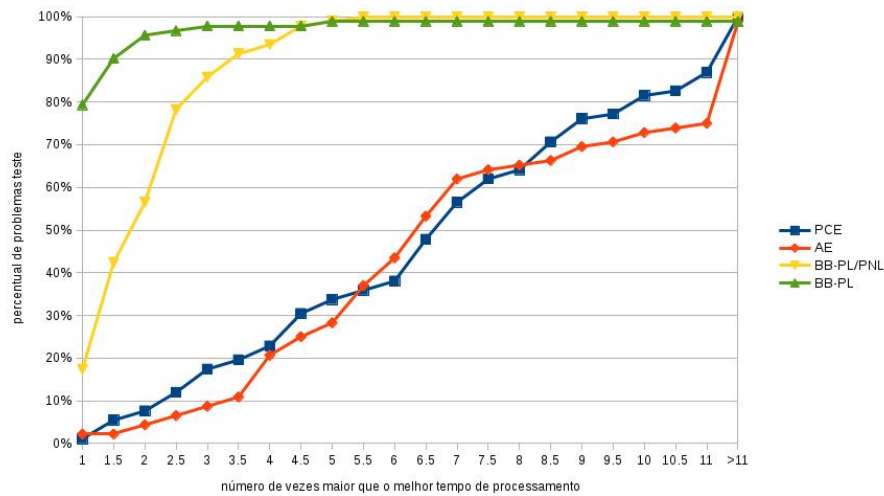


Figura 2: Comparação relativa de tempo de processamento (CPU) para os algoritmos nos 92 problemas com maior percentual de restrições não lineares.

Observando a Figura 2, pode-se perceber que, no subconjunto de instâncias de teste considerado, BB-PL apresentou menor tempo de processamento para cerca de 79% das instâncias de teste, enquanto que, para BB-PL/PNL, AE e PCE, esse número foi de cerca de 17%, 2% e 1%, respectivamente. Note que a diferença entre o resultado de BB-PL e BB-PL/PNL com relação a esse número é consideravelmente maior nesse subconjunto de instâncias. Uma possível explicação para essa diferença decorre do fato de que problemas com maior percentual de restrições não-lineares podem demandar maior esforço para o cálculo de suas funções, derivadas e sua resolução como um todo, o que também é corroborado pelo fato do desempenho de PCE ter se tornado mais competitivo ao de AE. É possível perceber ainda que, nesse subconjunto, BB-PL e AE foram capazes de resolver cerca de 99% das instâncias, ao passo que PCE e BB-PL/PNL resolveram 100%. O desempenho de BB-PL é levemente superado pelo de BB-PL/PNL apenas quando se consideram resultados superiores a 4,5 vezes o melhor tempo de processamento.

De um modo geral, os resultados apresentados pelas Figuras 1 e 2 indicam que BB-PL é uma abordagem bastante promissora ao se valer diretamente dos benefícios combinados da estratégia de geração de cortes de PCE, que é totalmente de primeira ordem, com o esquema avançado de aproximação linear de BB-PL/PNL, que tem gerado bons resultados na literatura. O desempenho de BB-PL dominou o de AE e PCE e BB-PL/PNL quando se considera tempo de processamento até 4 vezes maior que o melhor resultado nas instâncias em geral. Com respeito ao subconjunto de 92 instâncias com maior percentual de restrições não lineares, o desempenho de BB-PL foi ainda mais destacado, dominando as outras abordagens para tempo de processamento até 4,5 vezes maior que o melhor resultado, além de ter sido a abordagem mais eficiente em quase 80% das instâncias desse subconjunto. Além de ter se mostrado um método robusto nos experimentos realizados, BB-PL pode ser facilmente implementado através do uso de *lazy constraints* e funções *callback* em avançados pacotes de PLIM como **Cplex** e **Gurobi** sem exigir rotinas de PNL e cálculo de derivadas de segunda ordem.

Referências

- Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, and Nicolas Sawaya. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, May 2008. doi: 10.1016/j.disopt.2006.10.011.
- Brian Borchers and John E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programs. *Comput. Oper. Res.*, 21:359–367, April 1994. ISSN 0305-0548. doi: 10.1016/0305-0548(94)90024-8.
- CMU-IBM. Open source minlp project, <http://egon.cheme.cmu.edu/ibm/page.htm>, 2012.
- Marco Duran and Ignacio Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986. ISSN 0025-5610. 10.1007/BF02592064.
- Roger Fletcher and Sven Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994. ISSN 0025-5610. 10.1007/BF01581153.
- A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972. ISSN 0022-3239. 10.1007/BF00934810.
- Jan Kronqvist, Andreas Lundell, and Tapio Westerlund. The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. *Journal of Global Optimization*, 64(2):249–272, 2016. ISSN 1573-2916. doi: 10.1007/s10898-015-0322-3.
- Sven Leyffer. Integrating sqp and branch-and-bound for mixed integer nonlinear programming. *Comput. Optim. Appl.*, 18:295–309, March 2001. ISSN 0926-6003. doi: 10.1023/A:1011241421041.
- Sven Leyffer. Macminlp: Test problems for mixed integer nonlinear programming, 2003. <https://wiki.mcs.anl.gov/leyffer/index.php/macminlp>, 2013.
- Sven Leyffer, Jeff Linderoth, James Luedtke, Andrew Miller, and Todd Munson. Applications and algorithms for mixed integer nonlinear programming. *Journal of Physics: Conference Series*, 180(1):012014, 2009.
- Wendel Melo, Marcia Fampa, and Fernanda Raupp. Integrating nonlinear branch-and-bound and outer approximation for convex mixed integer nonlinear programming. *Journal of Global Optimization*, 60(2):373–389, 2014. ISSN 0925-5001. doi: 10.1007/s10898-014-0217-8.
- Wendel Melo, Marcia Fampa, and Fernanda Raupp. Algoritmo de plano de corte estendido modificado para programação não linear inteira mista. In *Anais do XLIX SBPO*, pages 2349–2360, 2017.
- Wendel Melo, Marcia Fampa, and Fernanda Raupp. An overview of minlp algorithms and their implementation in muriqui optimizer. *Submitted to Annals of Operations Research*, 2018.
- I. Quesada and I.E. Grossmann. An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers & Chemical Engineering*, 16(10-11):937 – 947, 1992. ISSN 0098-1354. doi: 10.1016/0098-1354(92)80028-8.
- Tapio Westerlund and Frank Pettersson. An extended cutting plane method for solving convex minlp problems. *Computers & Chemical Engineering*, 19, Supplement 1(0):131 – 136, 1995. ISSN 0098-1354. doi: 10.1016/0098-1354(95)87027-X.
- GAMS World. Minlp library 2, <http://www.gamsworld.org/minlp/minlplib2/html/>, 2014. URL <http://www.gamsworld.org/minlp/minlplib2/html/>.