Algoritmo de Plano de Corte Estendido Modificado para Programação Não Linear Inteira Mista

Wendel Melo
Faculdade de Computação
Universidade Federal de Uberlândia
wendelmelo@cos.ufrj.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, propomos uma modificação no algoritmo de Plano de Corte Estendido (PCE) para a resolução de problemas de programação não linear inteira mista convexa. Nossa abordagem, denominada como Plano de Corte Estendido Modificado (PCEM) é inspirada na estratégia de atualização do conjunto de pontos de linearização do algoritmo de Aproximação Externa (AE). Resultados computacionais sobre um conjunto de 343 instâncias de teste apontaram a efetividade do método proposto PCEM, cujo desempenho foi superior ao de PCE e competitivo ao de AE.

PALAVRAS CHAVE: Programação Não Linear Inteira Mista, Plano de Corte Estendido, Aproximação Externa.
Área de classificação principal: Otimização Combinatória.

ABSTRACT

In this work, we propose a modification on the Extended Cutting Plane algorithm (ECP) that solves mixed integer nonlinear programming problems. Our approach, called Modified Extended Cutting Plane (MECP), is inspired on the strategy of updating the set of linearization points in the Outer Approximation algorithm (OA). Computational results over a set of 343 test problems show the effectiveness of the proposed method MECP, which outperformes ECP and is competitive to OA.

KEYWORDS: Mixed Integer Nonlinear Programming, Extended Cutting Plane, Outer Approximation.
Main area: Combinatorial Optimization.

1 Introdução

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

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

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

Se, por um lado, a presença simultânea de variáveis contínuas e discretas juntamente com termos não lineares incorporados na função objetivo e/ou restrições tornam o problema (P) de difícil resolução, por outro lado, estes mesmos elementos fazem da PNLIM uma ferramenta bastante versátil para a aplicação em situações práticas diversas que envolvam algum tipo de otimização. Dentre as aplicações de PNLIM encontradas na literatura, podemos mencionar: planejamento de redes de telecomunicações [Belotti, 2009], agendamento de operações em refinarias de petróleo [Mouret and Grossmann, 2010], sistemas de geração de energia [Liu et al., 2009], projetos de cadeias de suprimentos [You and Grossmann, 2008], caminhos de robôs industriais [Gentilini, 2011], síntese de processos químicos [Duran and Grossmann, 1986], prevenção de conflitos de aeronaves [Christodoulou and Costoulakis, 2004], redes metabólicas em biotecnologia [Guillen and Pozo, 2010], redes de distribuição de água [Bragalli et al., 2012], trajetória de válvulas em séries de tanques em cascatas [Gopalakrishnan and Biegler, 2011], projeto de camada de isolante térmico para o Grande Colisor de Hádrons e recarga de núcleo de reatores nucleares [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; 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). A principal vantagem destes algoritmos é que eles se valem 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`.

Um algoritmo da classe de aproximação linear bastante difundido na literatura é o algoritmo de Aproximação Externa (AE). A cada iteração, AE resolve um problema de PLIM juntamente com um ou dois problemas de Programação Não Linear contínua (PNL). Tal esquema foi concebido de modo a garantir a convergência de AE em um número finito de iterações e tem se mostrado eficiente em muitos casos práticos. Um outro algoritmo da classe de aproximação linear também popular na literatura é o algoritmo de Plano de Corte Estendido (PCE). A principal diferença entre AE e PCE é que PCE não resolve qualquer problema de PNL ao longo de sua execução, se limitando apenas a resolver problemas de PLIM. Embora, a primeira vista, isto pareça ser uma vantagem de PCE, esta estratégia faz com que o mesmo não possua convergência garantida em número finito de iterações. Temos observado que, em muitos casos, o algoritmo PCE demanda um número de iterações maior

que AE para convergir para uma solução ótima do problema (P). Esta característica faz com que PCE necessite, em geral, de mais esforço computacional que AE. Todavia, por não precisar resolver problemas de PNL, o algoritmo PCE dispensa o cálculo de derivadas de segunda ordem, ou qualquer tipo de aproximação destas. Em algumas situações práticas, essa peculiaridade pode vir a se mostrar bastante vantajosa, especialmente em casos onde o cálculo de derivadas de segunda ordem é muito dispendioso ou não pode ser realizado por algum motivo.

Neste trabalho, propomos um algoritmo para PNLIM baseado em PCE. Nossa principal contribuição consiste em uma pequena modificação no algoritmo PCE de modo a torná-lo mais próximo à AE, na expectativa de reduzir assim o número de iterações necessárias para sua convergência, ao mesmo tempo em que mantemos a amigável propriedade de PCE de não necessitar solucionar qualquer tipo de problema de PNL, isentando assim o algoritmo do cálculo (ou da aproximação) de derivadas de segunda ordem. Apesar de modesta, nossa contribuição, denominada aqui como *Plano de Corte Estendido Modificado* (PCEM), apresentou resultados promissores para um conjunto de 343 problemas teste de PNLIM, indicando assim que o método proposto é competitivo em relação à AE.

Este texto está organizado da seguinte forma: a Seção 2 discute sobre o algoritmo PCE enquanto que o algoritmo AE é apresentado na Seção 3. Nossa abordagem PCEM é então introduzida na Seção 4. Por fim, a Seção 5 traz resultados computacionais comparando PCEM com AE e PCE, ao passo que a Seção 6 expõe algumas conclusões sobre o trabalho desenvolvido.

2 O Algoritmo de Plano de Corte Estendido

O algoritmo de Plano de Corte Estendido foi 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:

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

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 $(\bar{P})$:

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

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

Note que o ponto $(\bar{x},\bar{y})$ não precisa necessariamente satisfazer as restrições de (P). Note também 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 $L = \{(x^1,y^1),\ldots,(x^k,y^k)\}$ com k pontos de linearização que se constitui em uma relaxação do

problema original (P):

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

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.

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 **enquanto** $z^u - z^l > \epsilon_c$ **faça**
8 Seja $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ uma solução ótima de (M^L);
9 $z^l = \hat{\alpha}$;
10 **se** $(\hat{x}^k, \hat{y}^k)$ *é viável para* (P) *e* $f(\hat{x}^k, \hat{y}^k) < z^u$ **então**
11 $(x^*, y^*) = (\hat{x}^k, \hat{y}^k)$;
12 $z^u = f(\hat{x}^k, \hat{y}^k)$;
13 $L = L \cup (\hat{x}^k, \hat{y}^k)$;
14 $k = k + 1$;

Algoritmo 1: Algoritmo de Plano de Corte Estendido (PCE).

O algoritmo PCE é mostrado como Algoritmo 1. Ressaltamos que PCE não necessita solucionar qualquer tipo de problema de Programação Não Linear (PNL) e não trabalha com qualquer tipo de informação proveniente de derivadas de segunda ordem ou mesmo com aproximações destas. Essa característica pode se mostrar vantajosa em alguns casos, especialmente quando o cálculo de derivadas de segunda ordem é árduo ou não puder ser realizado por alguma circunstância qualquer. Apontamos ainda que a estratégia de acrescentar a solução $(\hat{x}, \hat{y})$ ao conjunto L ao final de cada iteração (linha 13) faz com que PCE não possua convergência finita. Temos observado ainda que os novos cortes gerados costumam ser fracos, o que leva o algoritmo a necessitar de um grande número de iterações para convergir para uma solução ótima. Na prática, este comportamento se mostra bastante desvantajoso em muitos casos, uma vez que, a cada iteração, um novo problema de PLIM (o problema mestre) precisa ser resolvido (linha 8). Ademais, a cada iteração, o número de restrições do problema mestre aumenta com o acréscimo de pontos ao conjunto L (linha 13), o que tende a aumentar o esforço necessário para sua resolução. Temos observado ainda

que, em muitas situações, algumas soluções $(\hat{\alpha}^k, \hat{x}^k, \hat{y}^k)$ subsequentes estão próximas umas das outras e o método PCE acaba gastando muitas iterações repetindo a mesma solução inteira $\hat{y}^k$.

3 O Algoritmo de Aproximação Externa

O algoritmo de 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 $\hat{y}^k$:

$$\begin{array}{lll} (P_{\hat{y}^k}) & \min\limits_{x} & f(x, \hat{y}^k) \\ & \text{s. a} & g(x, \hat{y}^k) \leq 0, \\ & & x \in X. \end{array} \tag{6}$$

Se $(P_{\hat{y}^k})$ for viável, seja $(\tilde{x}, \hat{y}^k)$ uma de suas soluções ótimas. Nesse caso, $f(\tilde{x}, \hat{y}^k)$ é um limite superior válido para (P) e $(\bar{P})$, e o ponto $(\tilde{x}, \hat{y}^k)$ é adicionado ao conjunto L. Observe que $(\tilde{x}, \hat{y}^k)$ 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:

$$\begin{array}{lll} (P^F_{\hat{y}^k}) & \min\limits_{u,x} & \sum_{i=1}^{m} u_i \\ & \text{s. a} & g(x, \hat{y}) \leq u, \\ & & u \geq 0, \\ & & x \in X,\ u \in \mathbb{R}^m. \end{array} \tag{7}$$

Seja $(\check{u}, \check{x})$ uma solução ótima de $(P^F_{\hat{y}^k})$ no contexto descrito. Então, o ponto $(\check{x}, \hat{y})$ é adicionado ao conjunto L. Após a atualização de L, com a inclusão de $(\tilde{x}, \hat{y})$ se o problema $(P_{\hat{y}^k})$ for viável, ou com a inclusão de $(\check{x}, \hat{y})$ 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 $(P^F_{\hat{y}^k})$, 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 despender 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. Adicionalmente, graças à sua boa performance, AE tem sido o algoritmo preferencial para a resolução de problemas convexos de PNLIM em diversos contextos. Todavia, é válido ressaltar que, quando o problema tratado é puramente discreto, isto é, não possui qualquer

variável contínua, AE e PCE tem desempenho fundamentalmente semelhante, uma vez que, nesta situação, os problemas $(P_{\hat{y}^k})$ e $(P^F_{\hat{y}^k})$ deixam de fazer sentido e, assumindo que ambos os algoritmos partem da mesma solução inicial (x^0, y^0), as soluções acrescentadas ao conjunto L seriam rigorosamente as mesmas.

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 $\left(M^L\right)$ 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 $(P^F_{\hat{y}^k})$ 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 $\left(M^L\right)$;
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 $(P^F_{\hat{y}^k})$;
19 $L = L \cup (x^k, \hat{y}^k)$;
20 $k = k + 1$;

Algoritmo 2: Algoritmo de Aproximação Externa (AE).

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 Nosso Algoritmo de Plano de Corte Estendido Modificado

Nesta seção, apresentamos nossa abordagem baseada no algoritmo PCE, aqui denominada como *Plano de Corte Estendido Modificado* (PCEM). Nossa principal motivação é melhorar o desempenho de PCE tornando-o mais próximo de AE, com a expectativa de assim diminuir o número de iterações necessárias para PCE convergir, ao mesmo tempo em que mantemos a característica de PCE de não resolver problemas de PNL e não utilizar qualquer informação proveniente de derivadas de segunda ordem ou mesmo de aproximações destas. Para isso, no lugar de considerar os problemas $(P_{\hat{y}^k})$ e $(P^F_{\hat{y}^k})$, PCEM considera uma aproximação linear para o problema $(P_{\hat{y}^k})$, construída sobre o mesmo conjunto de pontos de linearização L sobre o qual o problema (M^L) é construído. Denominamos então este novo problema como

$(M^L_{\hat{y}^k})$, definido a seguir:

$$\begin{array}{lll} \left(M^L_{\hat{y}^k}\right) & \min\limits_{\alpha,x} & \alpha \\ & \text{s. a} & \nabla f(x^j,y^j)^T \begin{pmatrix} x-x^j \\ \hat{y}^k-y^j \end{pmatrix} + f(x^j,y^j) \leq \alpha, \quad \forall (x^j,y^j) \in L, \\ & & \nabla g(x^j,y^j)^T \begin{pmatrix} x-x^j \\ \hat{y}^k-y^j \end{pmatrix} + g(x^j,y^j) \leq 0, \quad \forall (x^j,y^j) \in L, \\ & & x \in X. \end{array} \tag{8}$$

Observe que $(M^L_{\hat{y}^k})$ pode ser obtido a partir de (M^L) simplesmente fixando a variável y no valor $\hat{y}^k$. Desse modo, ao considerar o problema $(M^L_{\hat{y}^k})$, temos a expectativa de conseguir obter boas soluções viáveis de modo mais prematuro em comparação ao algoritmo PCE tradicional. Estas soluções são utilizadas para uma possível atualização do limite superior conhecido z^u e para fortalecer a relaxação provida pelo problema mestre através de seu acréscimo no conjunto L.

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 $(M^L_{\hat{y}^k})$ 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 $z^l = \hat{\alpha}$;
11 $L = L \cup (\hat{x}^k, \hat{y}^k)$;
12 **se** $(M^L_{\hat{y}^k})$ *é viável* **então**
13 Seja (α^k, x^k) uma solução ótima de $(M^L_{\hat{y}^k})$;
14 **se** $(x^k, \hat{y}^k)$ *é viável para* (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 3: Algoritmo de Plano de Corte Estendido Modificado (PCEM).

O algoritmo PCEM é exibido como Algoritmo 3. Em relação ao algoritmo PCE, a novidade aqui é a introdução das linhas 7, 12-17. Destacamos que, a cada iteração, dentre a resolução de (M^L) (linha 9) e a resolução de $(M^L_{\hat{y}^k})$ (linhas 12-13), a solução $(\hat{x}^k, \hat{y}^k)$ é acrescida ao conjunto L (linha 11), de modo a fortalecer a relaxação linear construída a partir dos pontos desse conjunto. Por essa razão, é possível que a solução ótima x^k de $(M^L_{\hat{y}^k})$ seja diferente de $\hat{x}^k$. Com esta estratégia, esperamos que o algoritmo PCEM encontre soluções viáveis mais rapidamente, e, deste modo, consiga fechar o *gap* de integralidade com menor esforço computacional em comparação a PCE. Observamos ainda que a solução obtida com a resolução de $(M^L_{\hat{y}^k})$ também é adicionada a L (linha 17), caso este seja viável. Uma vez que $(M^L_{\hat{y}^k})$ é um problema de programação linear, sua resolução não traz carga computacional significativa em comparação com a resolução de (M^L).

É válido ressaltar que a convergência de PCEM para a solução ótima de (P) é facilmente verificada a partir da convergência de PCE, uma vez que PCEM considera, ao longo de suas iterações, todos os pontos de linearização considerados por PCE (com alguns pontos adicionais). Apontamos ainda que, no contexto em discussão, não faria sentido a aproximação do problema de viabilidade $(P^F_{\hat{y}^k})$ em um problema de programação linear, pois caso $(M^L_{\hat{y}^k})$ seja inviável, o valor $\hat{y}^k$ já não mais poderá configurar como solução para y em (M^L) da iteração corrente em diante, e, portanto, não é necessária a resolução de qualquer problema de viabilidade para cortar de (M^L) soluções com o valor $\hat{y}^k$.

Por fim, frisamos que $(M^L_{\hat{y}^k})$ é uma tentativa de aproximar o problema $(P_{\hat{y}^k})$ usando programação linear. Desse modo, PCEM ainda é um método de primeira ordem que pode ser implementado sem qualquer rotina de resolução de problemas de programação não linear. Este último fato pode ser vantajoso em termos práticos, pois além das razões já mencionadas sobre dificuldade de cálculo de derivadas de segunda ordem em algumas aplicações, em muitos casos, mesmo as melhores rotinas computacionais de PNL podem falhar em convergir para a solução ótima do problema considerado, mesmo este problema sendo convexo e continuamente diferenciável. Em alguns casos, estas rotinas falham até mesmo para encontrar uma solução viável de problemas cuja região viável é não vazia, fazendo com que inviabilidade seja indevidamente declarada. Dessa forma, por não depender de rotinas de PNL, o algoritmo PCEM se mostra como uma abordagem robusta para a resolução de problemas diversos de PNLIM convexa.

5 Resultados Computacionais

Passamos aqui à apresentação de resultados computacionais oriundos da aplicação de PCE, AE e nossa abordagem PCEM 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 (%)	0,00	1,00	0,52	0,40
restrições	0	108217	1184,65	211
restrições lineares (%)	0,00	1,00	0,89	0,95

Os algoritmos supracitados 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 7.1.0`. Os testes rodaram em um computador com processador core i7 4790 (3,6 GHz), 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 significa dizer que os mesmos foram executados por um único processador a cada instante de tempo na máquina utilizada para os testes. O tempo de CPU de cada algoritmo em cada instância de teste foi limitado a 4 horas. Foram adotados os valores 10^{-6} como tolerância de convergência absoluta e 10^{-3} como tolerância de convergência relativa para todos os algoritmos.

A Figura 1 traz uma comparação relativa entre os algoritmos com respeito ao tempo de CPU dispendido no conjunto de instâncias de teste considerados. Note que os dados estão

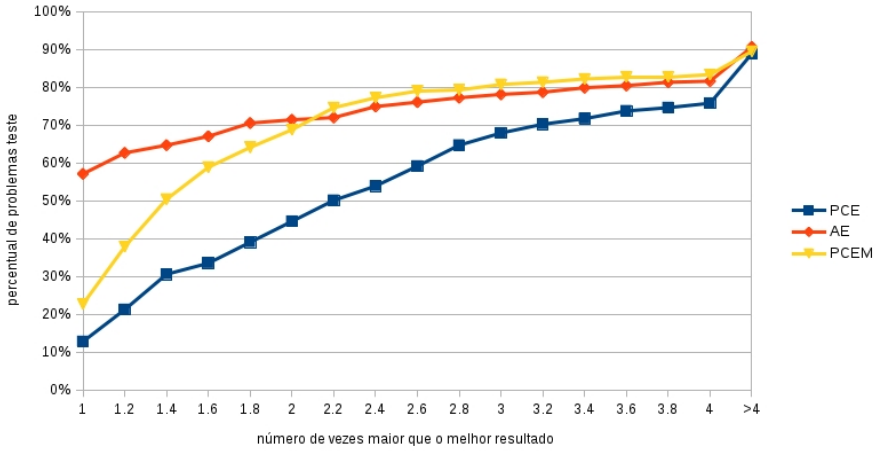

Figura 1: Comparação relativa de tempo de CPU para os algoritmos.

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 τ% das instâncias, o resultado obtido pelo algoritmo em questão é menor ou igual a α vezes o melhor tempo computacional dentre todos os algoritmos.

Note, por exemplo, que a curva de AE passa pelo ponto (1, 57%). Isto significa que, para 57% 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 (1,2, 63%), indicando que AE conseguiu resolver 63% das instâncias gastando até 20% a mais de tempo que o melhor algoritmo em cada instância (1,2 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.

Analisando a Figura 1, podemos observar que o desempenho de AE domina o de PCE. É possível notar também que nosso algoritmo PCEM apresentou resultados substancialmente melhores que PCE, dominando completamente o seu desempenho e chegando até a se tornar competitivo em relação ao algoritmo AE. É válido destacar que a curva de PCEM domina, ainda que de forma modesta, a de AE para resultados maiores ou iguais a 2,2 vezes o melhor resultado. Todos os algoritmos foram capazes de resolver cerca de 90% das instâncias de teste no tempo máximo de execução estipulado.

A seguir, avaliamos o desempenho dos algoritmos sobre os problemas teste com menor percentual de restrições lineares, isto é, sobre os problemas com maior percentual de restrições não lineares. Separamos então todas as 92 instâncias do grupo de 343 com até 80% de restrições lineares para gerar um novo gráfico comparativo entre as abordagens com relação ao tempo computacional, exibido na Figura 2.

Examinando a Figura 2, observamos que, para os problemas onde o percentual de restrições lineares é menor, AE não conseguiu exibir o mesmo sucesso apresentado para todo o grupo de problemas teste, sendo superado de forma substancial pelas demais abordagens. Novamente, fica evidente que nossa abordagem PCEM dominou o desempenho de PCE, sendo nesse contexto, indiscutivelmente o algoritmo de melhor performance.

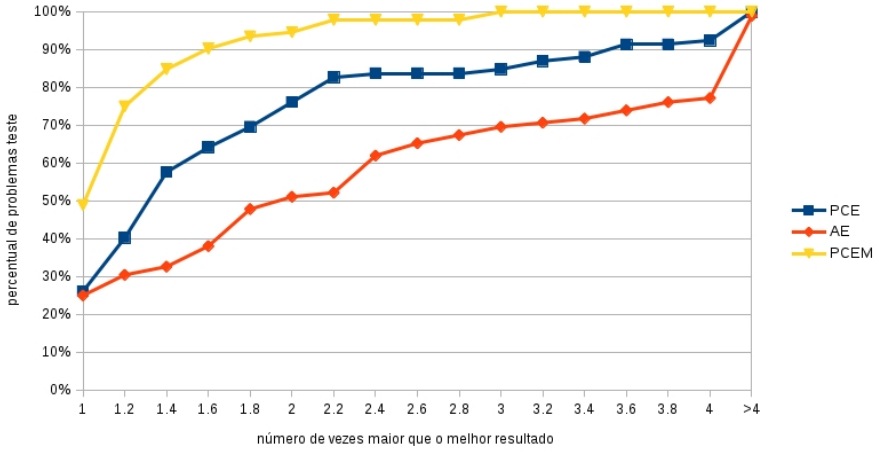

Figura 2: Comparação relativa de tempo de CPU para os algoritmos.

Por fim, a Tabela 2 apresenta informações sobre a razão de resultados entre PCEM e PCE com respeito a tempo de CPU e número de iterações para o grupo total de 343 instâncias de teste e para o grupo de 92 instâncias com menor percentual de restrições lineares. É possível observar, por exemplo, que, para o grupo total de 343 instâncias, PCEM demandou, em média, cerca de 77,2% do tempo computacional demandado por PCE, enquanto que para as instâncias de menor percentual de restrições lineares, esse valor foi de cerca de 84,3%.

Tabela 2: Informações sobre a razão entre resultados de PCEM e PCE.

	Todos		**Até 80% de rest lin**	
	tempo de CPU	**#iterações**	**tempo de CPU**	**#iterações**
média	0,772	0,719	0,843	0,823
mediana	0,720	0,647	0,816	0,852
min	0,029	0,328	0,029	0,400
max	2,082	1,364	2,082	1,364

6 Conclusões

Uma modificação no algoritmo de Plano de Corte Estendido (PCE) foi proposta neste trabalho para a resolução de problemas de Programação Não Linear Inteira Mista (PNLIM) convexa. Nossa abordagem, denominada como *Plano de Corte Estendido Modificado* (PCEM) foi construída de modo a se tornar mais próxima ao conhecido algoritmo de Aproximação Externa (AE), mantendo a peculiaridade de PCE de não fazer uso de informação proveniente de derivadas de segunda ordem ou de aproximação destas. Resultados computacionais sobre um vasto conjunto de instâncias de teste indicaram que o algoritmo PCEM é robusto e promissor em comparação com PCE e AE. De modo geral, PCEM dominou a performance de PCE para o conjunto total de instâncias de teste e para o seleto subgrupo com as instâncias de maior percentual de restrições não lineares. Em relação a AE, PCEM foi competitivo neste primeiro grupo de instâncias, e indiscutivelmente superior no subgrupo seleto, onde AE ainda foi superado por PCE. Implementações de todos os algoritmos considerados nesse estudo, PCE, PCEM e AE, juntamente com heurísticas [Melo et al., 2015,

2016] estarão disponíveis em um novo pacote de PNLIM aberto ainda em desenvolvimento, o *solver* `Muriqui`.

Referências

Pietro Belotti. Design of telecommunication networks with shared protection, Modification of: 18:24:11, August 31 2009. Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=51`.

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.

Cristiana Bragalli, Claudia D'Ambrosio, Jon Lee, Andrea Lodi, and Paolo Toth. On the optimal design of water distribution networks: a practical minlp approach. *Optimization and Engineering*, 13:219–246, 2012. ISSN 1389-4420. 10.1007/s11081-011-9141-7.

M. Christodoulou and C. Costoulakis. Nonlinear mixed integer programming for aircraft collision avoidance in free flight. In *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, volume 1, pages 327 – 330 Vol.1, may 2004. doi: 10.1109/MELCON.2004.1346858.

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.

Iacopo Gentilini. Minlp approach for the tspn (traveling salesman problem with neighborhoods), Modification of: 15:58:23, April 15 2011. Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=124`.

A. M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972. ISSN 0022-3239. 10.1007/BF00934810.

Ajit Gopalakrishnan and Lorenz Biegler. Minlp and mpcc formulations for the cascading tanks problem, Modification of: 14:14:10, November 30 2011. Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=140`.

Gonzalo Guillen and Carlos Pozo. Optimization of metabolic networks in biotechnology, Modification of: 05:15:39, February 19 2010. Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=81`.

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. URL `https://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP`.

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.

Pei Liu, Efstratios N. Pistikopoulos, and Zheng Li. Global multi-objective optimization of a nonconvex minlp problem and its application on polygeneration energy systems design, Modification of: 06:13:12, July 30 2009. Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=42`.

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. Um novo algoritmo de minimização de gap para programação não linear inteira mista binária. In *Anais do XLVII SBPO*, pages 2715–2726, sep 2015.

Wendel Melo, Marcia Fampa, and Fernanda Raupp. An integrality gap minimization heuristic for binary mixed integer nonlinear programming. In *Proceedings of the XIII Global Optimization Workshop: GOW'16*, pages 73–76, 2016.

Sylvain Mouret and Ignacio Grossmann. Crude-oil operations scheduling, Modification of: 05:03:10, November 21 2010. Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=117`.

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. An International Journal of Computer Applications in Chemical Engineering.

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. European Symposium on Computer Aided Process Engineering.

GAMS World. Minlp library 2, http://www.gamsworld.org/minlp/minlplib2/html/, 2014. URL `http://www.gamsworld.org/minlp/minlplib2/html/`.

Fengqi You and Ignacio E. Grossmann. Mixed-integer nonlinear programming models and algorithms for large-scale supply chain design with stochastic inventory management. *Industrial & Engineering Chemistry Research*, 47(20):7802–7817, 2008. doi: 10.1021/ie800257x.