Um Novo Algoritmo de Minimização de *Gap* para Programação Não Linear Inteira Mista Binária

Wendel Melo
Instituto Alberto Luiz Coimbra
de Pós-graduação e Pesquisa em Engenharia (COPPE)
Universidade Federal do Rio de Janeiro
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

Este trabalho apresenta um novo algoritmo para Programação Não Linear Inteira Mista binária baseado na ideia de minimização do *gap* de integralidade incorporada a um procedimento de *branch-and-bound*. Nossa proposta, baseada no trabalho de Melo et al. (2012), supera a principal desvantagem deste último, que consiste na resolução de problemas não convexos até a otimalidade global. Mostramos ainda que existe uma relação evidente entre nossa abordagem e um algoritmo padrão de *branch-and-bound* não linear, com a vantagem de que, em nosso método, soluções viáveis tendem a ser mais rapidamente encontradas. Esta última característica possibilita o seu uso na qualidade de heurística de obtenção de soluções viáveis de forma bastante competitiva.

PALAVRAS CHAVE: Programação Não Linear Inteira Mista, *Branch-And-Bound*, Minimização de *Gap* de Integralidade.
Área de classificação principal: Otimização Combinatória

ABSTRACT

This work presents a new algorithm for binary Mixed Integer Nonlinear Programming based on the idea of minimizing the integrality gap, incorporating it to a branch-and-bound procedure. Our proposal is based on Melo et al. (2012) and overcomes the main drawback from this last, since we do not require global optimality of non-convex problems. Moreover, we show that there is a clear relation between our approach and a standard nonlinear branch-and-bound algorithm. Our method still has the advantage of finding feasible solutions quickly, and this feature enables its usage as a heuristic procedure in a very competitive way.

KEYWORDS: Mixed Integer Nonlinear Programming, Branch-And-Bound, Minimization of Integrality Gap.
Main area: Combinatorial Optimization

1 Introdução

Problemas de Programação Não Linear Inteira Mista (PNLIM) são caracterizados pela presença simultânea de variáveis em domínios discretos e expressões não lineares incorporadas na função objetivo e/ou restrições. Particularmente, abordamos nesse trabalho o seguinte problema de PNLIM onde todas as variáveis inteiras são binárias, isto é, restritas aos valores $\{0, 1\}$:

$$\begin{array}{lll} (P) & \text{minimizar}_{x,y} & f(x,y) \\ & \text{sujeito a:} & g(x,y) \leq 0, \\ & & x \in X,\ y \in Y \cap \{0,\ 1\}^{n_y}, \end{array} \tag{1}$$

onde X e Y são subconjuntos poliédricos de $\mathbb{R}^{n_x}$ e $\mathbb{R}^{n_y}$, respectivamente. As funções $f : X \times Y \to \mathbb{R}$ e $g : X \times Y \to \mathbb{R}^m$ são convexas, contínuas e duplamente diferenciáveis.

PNLIM tem sido objeto de estudo de diversos pesquisadores em tempos recentes. O leitor interessado pode conferir bons trabalhos de revisão bibliográfica em Bonami et al. (2009b), D'Ambrosio and Lodi (2011), Hemmecke et al. (2010), Melo (2012), Trespalacios and Grossmann (2014). Ao longo dos últimos anos, diversos algoritmos foram apresentados para a resolução de problemas de PNLIM. Nós os classificamos aqui nas seguintes categorias:

1. **Algoritmos de aproximação linear:** Este tipo de algoritmo resolve um problema de PNLIM convexo aproximando-o por uma sequência problemas de Programação Linear Inteira Mista (PLIM), cujas resoluções fornecem limites inferiores (no caso de minimização) para o problema abordado. A principal vantagem destes algoritmos é que eles se valem de todo o avanço e maturidade já obtidos para a área de PLIM. Entretanto, a cada iteração, um novo problema de PLIM com mais linearizações que o anterior precisa ser resolvido, o que, em alguns casos, pode tornar estes algoritmos desvantajosos em comparação com os demais. Os principais representantes dessa classe de algoritmo são os algoritmos de decomposição generalizada de Benders (Geoffrion (1972)), aproximação externa de Duran and Grossmann (1986) e Fletcher and Leyffer (1994), branch-and-bound linear/não linear baseado em aproximação externa de Quesada and Grossmann (1992) e plano de corte estendido de Westerlund and Pettersson (1995).

2. **Algoritmos de *branch-and-bound* não linear:** De modo geral, esta classe estende ao contexto não linear a conhecida metodologia de particionamento de espaço conhecida como *Branch-and-Bound* (BB), já amplamente difundida no universo de PLIM. O sucesso na aplicação de BB para PLIM está fortemente relacionado à adoção de técnicas para aceleração de sua convergência tais como pré-processamento, início promissor (*hot start*) e geração de cortes. Estas técnicas não são facilmente estendidas ao universo não linear, o que, aliado à dificuldade na resolução de problemas de Programação Não Linear contínua (PNL), faz com que os algoritmos de BB ainda não consigam repetir neste novo ambiente o mesmo êxito alcançado em PLIM. Trabalhos envolvendo algoritmos dessa classe para PNLIM podem ser conferidos em Bonami et al. (2011), Borchers and Mitchell (1994), Gupta and Ravindran (1985), Leyffer (2001), Still and Westerlund (2006), Stubbs and Mehrotra (1999).

3. **Algoritmos de subproblemas contínuos:** Algoritmos desta categoria são caracterizados pela resolução de uma sequência de problemas em domínio contínuo. Diferentemente dos algoritmos de BB, esta classe não faz particionamento explícito do espaço em seu processo de convergência. Como representantes desta classe, podemos citar algoritmos baseados em programação quadrática sequencial (Exler and Schittkowski

(2007), Exler et al. (2012)), o trabalho de Murray and Ng (2010), onde o problema de PNLIM binário com restrições lineares é reformulado como um problema não-convexo cuja resolução é feita através de um esquema de penalização e suavização, e, por fim, Melo et al. (2012) que apresentam um algoritmo baseado na resolução sequencial de problemas minimizadores do *gap* de integralidade. Este último trabalho foi tomado por base para a elaboração do método aqui proposto.

4. **Reformulação como problema de programação disjuntiva:** Nos últimos anos, alguns trabalhos têm levantado a questão de que problemas de PNLIM podem ser reformulados como problemas de Programação Disjuntiva (PD), que são problemas de otimização que incorporam variáveis lógicas para decidir se determinados conjuntos de restrições devem ser satisfeitos ou não. Nesse contexto, problemas de PNLIM podem então ser tratados por algoritmos desenvolvidos no âmbito de PD. O leitor interessado no assunto pode consultar os recentes trabalhos de Grossmann and Lee (2003), Grossmann and Ruiz (2012), Lee and Grossmann (2000), Raman and Grossmann (1994), Trespalacios and Grossmann (2014).

5. **Abordagens híbridas:** Os algoritmos dessa categoria procuram combinar características presentes nas categorias anteriores, ou ainda, combinar algoritmos dentro de uma mesma categoria. O objetivo principal é aproveitar as principais vantagens das abordagens misturadas e, de forma simultânea, remediar seus pontos fracos. Como representantes dessa classe, podemos mencionar o algoritmo de Bonami et al. (2008), que mistura o algoritmo de BB linear/não linear baseado em aproximação externa com o algoritmo de aproximação externa puro, e Melo et al. (2014), que mistura branch-and-bound não linear padrão com aproximação externa.

Visando a resolução de (P), Melo et al. (2012) apresentaram um algoritmo de subproblemas contínuos denominado *Heurística de Minimização do Gap de Integralidade*, ou HMGI, que é apoiado na resolução de problemas não convexos de minimização de *gap* de integralidade. Embora os autores o tenham cautelosamente denominado como heurística, HMGI é capaz de convergir para a solução ótima do problema abordado se otimalidade global for garantida na resolução dos seus problemas de minimização de *gap*. Embasados nesse trabalho, introduzimos aqui um novo algoritmo para PNLIM, ao qual chamamos *Algoritmo de Minimização do Gap de Integralidade 2*, ou AMGI2. A abordagem desenvolvida adota a ideia de minimização de *gap* integrada a um procedimento de *branch-and-bound*. A principal vantagem em relação à HMGI, é que AMGI2 pode convergir para a solução ótima do problema tratado mesmo quando otimalidade global não é garantida na resolução dos problemas de minimização de *gap*, o que o classifica inquestionavelmente como sendo um algoritmo exato, isto é, capaz de convergir, com comprovação, para a solução ótima do problema abordado.

Pode-se identificar uma relação clara entre AMGI2 e um algoritmo padrão de *branch-and-bound* não linear, com a vantagem de que AMGI2 possui foco em encontrar soluções viáveis rapidamente. Esta habilidade pode lhe render vantagens práticas conforme será discutido adiante, além de tornar interessante a adoção de AMGI2 na qualidade de heurística, bastando para isso a interrupção precoce do algoritmo visando apenas a obtenção de solução viável.

A apresentação do algoritmo aqui proposto está estruturada em 5 seções. A Seção 2 detalha o algoritmo HMGI de Melo et al. (2012). A Seção 3 descreve nosso algoritmo AMGI2. Resultados preliminares em que AMGI2 é avaliado como heurística são apresentados na Seção 4. Por fim, a Seção 5 traz conclusões e perspectivas futuras.

2 Heurística de Minimização do *Gap* de Integralidade

Melo et al. (2012) propuseram a *Heurística de Minimização do Gap de Integralidade*, ou HMGI. A ideia principal da abordagem é a reformulação de (P) como o seguinte problema de Programação Não Linear contínuo (PNL):

$$
\begin{aligned}
(\bar{P}) \quad \text{minimizar}_{x,y} \quad & f(x,y) && (2)\\
\text{sujeito a:} \quad & g(x,y) \leq 0, && (3)\\
& \sum_{i=1}^{n_y} y_i(1-y_i) = 0, && (4)\\
& 0 \leq y \leq 1, && (5)\\
& x \in X,\ y \in Y. && (6)
\end{aligned}
$$

Observe que a restrição de integralidade das variáveis em (P) foi substituída pelas restrições (4) e (5). Note ainda que, dada uma solução qualquer $(\hat{x}, \hat{y})$ para (P), o lado esquerdo da expressão (4) nos fornece uma medida do *gap* de integralidade de $(\hat{x}, \hat{y})$, isto é, uma medida do quão distante $\hat{y}$ está de satisfazer a restrição de integralidade. Embora o problema $(\bar{P})$ seja contínuo, a restrição (4) o torna não convexo com a região viável desconexa, o que dificulta acentuadamente a sua resolução por parte de pacotes de PNL. Murray and Ng (2010) também consideraram as restrições (4) e (5) para reformular um problema de PNLIM binário com restrições lineares. Os mesmos trataram a inconveniência da restrição (4) penalizando-a na função objetivo e resolvendo o problema resultante com um esquema de suavização.

Os autores de HMGI optaram por construir um problema auxiliar contínuo (P^G) a partir de (P), onde o objetivo é a minimização do lado direito de (4). Por esta razão, os autores o denominaram como *problema de minimização do gap de integralidade*:

$$
\begin{aligned}
\left(P^G\right) \quad \text{minimizar}_{x,y} \quad & \sum_{i=1}^{n_y} y_i(1-y_i) && (7)\\
\text{sujeito a:} \quad & g(x,y) \leq 0, && (8)\\
& 0 \leq y \leq 1, && (9)\\
& x \in X,\ y \in Y. && (10)
\end{aligned}
$$

Embora a função objetivo de (P^G) seja côncava (não convexa), sua região viável ainda permanece convexa, o que favorece a obtenção de soluções por parte dos pacotes de PNL em geral. É facilmente verificável que qualquer solução viável de (P) é ótima global de (P^G), e de posse dessa informação, o problema (P^G) é resolvido em busca de uma solução viável $(\bar{x}, \bar{y})$ para (P). Após esse processo, a solução $(\bar{x}, \bar{y})$ é utilizada para a construção de um subproblema de (P), onde as variáveis y são fixadas no valor $\bar{y}$:

$$
\begin{aligned}
(P(\bar{y})) \quad \text{minimizar}_{x} \quad & f(x,\bar{y}) && \\
\text{sujeito a:} \quad & g(x,\bar{y}) \leq 0, && (11)\\
& x \in X. &&
\end{aligned}
$$

Vale ressaltar que a estratégia de resolução do problema $(P(\bar{y}))$ é utilizada em alguns algoritmos de aproximação linear visando a melhoria de soluções encontradas e obtenção de melhores cortes lineares. Observe ainda que a resolução do problema $(P(\bar{y}))$ fornece uma solução $\tilde{x}$ tal que $(\tilde{x}, \bar{y})$ é a melhor solução para (P) tendo $\bar{y}$ como valor para a variável inteira y. Essa solução é então utilizada para atualizar o limite superior do algoritmo HMGI. De posse do limite superior válido $z^u = f(\tilde{x}, \bar{y})$, HMGI passa a então a considerar,

iterativamente, o seguinte problema denominado como *problema de minimização de gap de integralidade com corte de nível objetivo*:

$$(\hat{P}^G(z^u, \epsilon_c)) \quad \text{minimizar}_{x,y} \quad \sum_{i=1}^{n_y} y_i(1-y_i) \tag{12}$$

$$\text{sujeito a:} \quad g(x,y) \leq 0, \tag{13}$$

$$f(x,y) \leq z^u - \epsilon_c, \tag{14}$$

$$0 \leq y \leq 1, \tag{15}$$

$$x \in X,\ y \in Y, \tag{16}$$

onde z^u é o melhor limite superior obtido para (P), e $\epsilon_c > 0$ é uma tolerância de convergência. Note que a única diferença entre (P^G) e $(\hat{P}^G(z^u, \epsilon_c))$ é que este último problema traz a restrição de corte de nível objetivo (14), que visa a melhoria do limite superior conhecido, quando possível.

Tendo introduzido as formulações consideradas por HMGI, estamos prontos a apresentar o algoritmo HMGI (Algoritmo 1). O Teorema 1 trata da convergência de HMGI para a solução ótima do problema abordado.

Entrada: (P): problema de PNLIM abordado, ϵ_c: tolerância de convergência
Saída: (x^*, y^*): Solução ótima para (P)

```
Seja (x̄^0, ȳ^0) uma solução ótima global de (P^G) ;
se ȳ^0 for inteira então
    Seja x̃^0 uma solução ótima de (P(ȳ^0)) ;
    (x*, y*) = (x̃^0, ȳ^0) ;
    z^u = f(x̃^0, ȳ^0) ;
    k = 1 ;
    enquanto (P̂^G(z^u, ε_c)) for viável faça
        Seja (x̄^k, ȳ^k) uma solução ótima global de P̂^G(z^u, ε_c) ;
        se ȳ^k for inteira então
            Seja x̃^k uma solução ótima de P(ȳ^k) ;
            (x*, y*) = (x̃^k, ȳ^k) ;
            z^u = f(x̃^k, ȳ^k) ;
        senão
            Pare o algoritmo ;
        k = k + 1 ;
```

Algoritmo 1: Algoritmo HMGI.

Teorema 1 *Sejam (P^G) e $(\hat{P}^G(z^u, \epsilon_c))$ os problemas de minimização de gap de integralidade tomados pelo algoritmo HMGI (Algoritmo 1). Se estes problemas sempre puderem ser resolvidos até a sua otimalidade global ao longo da execução deste algoritmo, então o mesmo convergirá para a solução ótima de (P), com tolerância de ϵ_c, em um número finito de iterações.*

A demonstração do Teorema 1 dada em Melo et al. (2012) se baseia no fato de que a cada iteração k, o algoritmo HMGI visita uma solução inteira $\bar{y}^k$ diferente. Uma vez que o problema é binário, o número de soluções inteiras é finito e, assim, o algoritmo HMGI converge em um número finito de iterações nas condições estabelecidas.

3 Algoritmo de Minimização do *Gap* de Integralidade 2

Conforme o Teorema 1 enuncia, a convergência para a otimalidade do algoritmo HMGI está totalmente fundamentada na obtenção de soluções ótimas globais para os problemas de minimização de *gap*. Embora esses problemas não convexos possuam particularidades que poderiam facilitar a sua resolução, como, por exemplo, região viável convexa, função objetivo quadrática côncava separável e limite inferior conhecido, o processo prático de resolução desses problemas pode ainda se mostrar bastante árduo. Essa desvantagem é agravada pelo fato de que uma série de problemas desse tipo precisa ser resolvida e que pode ser difícil a obtenção de uma rotina computacional eficiente para desempenhar essa tarefa. Com estas informações em mente, desenvolvemos um novo algoritmo baseado na ideia de minimização de *gap* de HMGI integrada a um procedimento de *branch-and-bound*. Este novo algoritmo não exige otimalidade global na resolução dos problemas de minimização de *gap*, o que lhe permite ser implementado com alguma das diversas rotinas computacionais de Programação Não Linear (PNL) ao qual dispomos nos dias de hoje. Denominamos nossa abordagem como *Algoritmo de Minimização do Gap de Integralidade 2*, ou AMGI2.

Segundo nosso melhor conhecimento, uma das metodologias mais efetivas para a resolução de problemas não convexos é o algoritmo de *Branch-and-Bound* Espacial (BBE) (O leitor interessado pode consultar Adjiman et al. (1998)). Considerando a aplicação de um algoritmo de BBE básico sobre um dos problemas de minimização de *gap*, no momento apropriado para a ramificação (*branching*), seria necessária a escolha de uma variável y_p para a geração de duas novas subdivisões. Em uma dessas subdivisões, seria adotada a restrição $y_p \leq \upsilon$, enquanto que na outra seria adotada a restrição $y_p \geq \upsilon$, onde υ seria um número real escolhido no intervalo $(\bar{l}_{y_p},\ \bar{u}_{y_p})$, onde $\bar{l}_{y_p}$ e $\bar{u}_{y_p}$ seriam, respectivamente, os limites inferior e superior de y_p na subdivisão do espaço corrente (inicialmente, $\bar{l}_{y_p} = 0$ e $\bar{u}_{y_p} = 1$). Embora um algoritmo de BBE básico divida o espaço das variáveis sobre valores υ fracionários, já sabemos de antemão que as soluções de interesse para os problemas de minimização de *gap* apresentam valores inteiros para as variáveis y, pois essas são as soluções que minimizam nossa função objetivo que mede o *gap* de integralidade. Levando em conta essa valiosa informação, construímos um nova abordagem integrando o Algoritmo 1 a um procedimento de *branch-and-bound* que, em seu processo de ramificação, divide o espaço tomando por base os valores inteiros que as variáveis y podem assumir, de modo similar ao *branch-and-bound* usado em programação inteira.

Definimos então o problema de minimização de *gap* de integralidade com corte de nível objetivo tomado em um determinado nó k da enumeração de *branch-and-bound*:

$$
\begin{aligned}
(\hat{P}^G(Y^k, z^u, \epsilon_c)) \quad \text{minimizar}_{x,y} \quad & \sum_{i=1}^{n_y} y_i(1-y_i) && (17)\\
\text{sujeito a:} \quad & g(x,y) \ \leq \ 0 && (18)\\
& f(x,y) \ \leq \ z^u - \epsilon_c && (19)\\
& 0 \leq y \leq 1 && (20)\\
& x \in X,\ y \in Y^k && (21)
\end{aligned}
$$

onde Y^k é a partição de Y referente ao nó k da árvore de *branch-and-bound*. AMGI2 foi projetado para ser utilizado com pacotes de PNL que podem convergir para soluções ótimas locais ao lidarem com os problemas de minimização de *gap* considerados. A cada vez que uma solução ótima (possivelmente local) $(\bar{x}, \bar{y})$ com $\bar{y}$ não inteiro é obtida para um desses problemas, particionamos o espaço ramificando sobre alguma variável y_j com valor $\bar{y}_j$ fracionário. Por outro lado, quando obtemos $\bar{y}$ inteiro nessas mesmas circunstâncias,

utilizamos essa nova solução para obter, a partir de $P(\bar{y})$, a solução $(\tilde{x}, \bar{y})$ que atualizará o limite superior z^u. De posse dessa última, resolvemos novamente o problema de minimização de *gap* na mesma partição corrente, mas agora com a restrição de corte de nível objetivo atualizada com o novo limite superior z^u. Repetimos esses passos na partição corrente até obter uma solução $(\bar{x}, \bar{y})$ com $\bar{y}$ não inteiro para $(\hat{P}^G(Y^0, z^u, \epsilon_c))$, ou este problema se tornar inviável. Nesse primeiro caso ($\bar{y}$ não inteiro), procedemos à ramificação. Nesse segundo caso ($(\hat{P}^G(Y^0, z^u, \epsilon_c))$ inviável), realizamos poda.

Entrada: (P): problema de PNLIM abordado, ϵ_c: tolerância de convergência
Saída: (x^*, y^*): Solução ótima de (P)

1 $Y^0 = Y$;
2 $z^u = \infty$;
3 Seja $(\bar{x}^0, \bar{y}^0)$ uma solução ótima (possivelmente local) de (P^G) ;
4 **enquanto** $\bar{y}^0$ *for inteira* **faça**
5 Seja $\tilde{x}^0$ uma solução ótima de $(P(\bar{y}^0))$;
6 $(x^*, y^*) = (\tilde{x}^0, \bar{y}^0)$;
7 $z^u = f(\tilde{x}^0, \bar{y}^0)$;
8 **se** $(\hat{P}^G(Y^k, z^u, \epsilon_c))$ *é inviável* **então**
9 **Pare o algoritmo** ;
10 Seja $(\bar{x}^0, \bar{y}^0)$ uma solução ótima (possivelmente local) de $(\hat{P}^G(Y^0, z^u, \epsilon_c))$;
11 Selecione uma variável y_j com valor $\bar{y}^0_j$ não inteiro ;
12 $Y^1 = Y \cap \{y \in \mathbb{R}^{n_y} : y_j = 0\}$;
13 $Y^2 = Y \cap \{y \in \mathbb{R}^{n_y} : y_j = 1\}$;
14 Seja $N = \{1, 2\}$ a lista inicial de nós em aberto ;
15 $i = 2$;
16 **enquanto** $N \neq \emptyset$ **faça**
17 Selecione um nó k de N ;
18 **se** $(\hat{P}^G(Y^k, z^u, \epsilon_c))$ *é viável* **então**
19 Seja $(\bar{x}^k, \bar{y}^k)$ uma solução ótima (possivelmente local) de $(\hat{P}^G(Y^k, z^u, \epsilon_c))$;
20 **enquanto** $\bar{y}^k$ *for inteira* **faça**
21 Seja $\tilde{x}^k$ uma solução ótima de $(P(\bar{y}^k))$;
22 $(x^*, y^*) = (\tilde{x}^k, \bar{y}^k)$;
23 $z^u = f(\tilde{x}^k, \bar{y}^k)$;
24 **se** $(\hat{P}^G(Y^k, z^u, \epsilon_c))$ *é inviável* **então**
25 **Vá para a linha 32** ;
26 Seja $(\bar{x}^k, \bar{y}^k)$ uma solução ótima (possivelmente local) de $(\hat{P}^G(Y^k, z^u, \epsilon_c))$;
 `// Ramificação`
27 Selecione uma variável y_j com valor $\bar{y}^k_j$ não inteiro ;
28 $Y^{i+1} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 0\}$;
29 $Y^{i+2} = Y^k \cap \{y \in \mathbb{R}^{n_y} : y_j = 1\}$;
30 $N = N \cup \{i+1, i+2\}$;
31 $i = i + 2$;
32 $N = N \backslash \{k\}$;

Algoritmo 2: Algoritmo AMGI2.

O algoritmo de AMGI2 é mostrado como Algoritmo 2. Observe que um procedimento único de *branch-and-bound* é utilizado de forma integrada às resoluções dos problemas de minimização de *gap*. Uma estratégia similar foi utilizada por Quesada and Grossmann (1992) para apresentar um algoritmo para PNLIM baseado no algoritmo de aproximação externa de Duran and Grossmann (1986), que originalmente pode se valer de várias aplicações

sequenciais de *branch-and-bound* para a resolução de problemas de PLIM aproximativos.

Ressaltamos que no procedimento de *branch-and-bound* em AMGI2, não dispomos de qualquer artifício para obtenção de limites inferiores para cada partição. Por esta razão, podas por limite não podem ser realizadas. Podas por otimalidade das partições também se mostram impraticáveis, pois a obtenção de uma solução inteira em uma partição não garante a inexistência de soluções inteiras melhores nessa mesma partição. Dessa forma, o algoritmo AMGI2 só pode eliminar partições por meio de poda por inviabilidade. Observe, entretanto, que a restrição de corte de nível objetivo (19) provoca inviabilidade em determinadas partições que não possam melhorar o limite superior, isto é, partições que seriam podadas por limite em um *branch-and-bound* tradicional de programação inteira são podadas por inviabilidade em nossa abordagem. De forma similar, quando a melhor solução inteira de uma partição por ventura é obtida, esta mesma restrição causará inviabilidade na partição corrente ou em partições descendentes. Assim, podas que ocorreriam por otimalidade em um *branch-and-bound* tradicional de programação inteira ocorrem também por inviabilidade em nossa abordagem.

Há uma relação evidente entre o algoritmo de *branch-and-bound* não linear e AMGI2. A principal diferença diz respeito ao fato que um *branch-and-bound* tradicional otimiza em cada partição observando somente a função objetivo original do problema tratado, o que pode levar à soluções nas partições que estejam longe da integralidade. Uma consequência bem conhecida deste fato é que muitas vezes um algoritmo de *branch-and-bound* pode demandar muitas iterações até encontrar alguma solução viável para o problema abordado. AMGI2 por sua vez otimiza nas partições buscando encontrar alguma solução inteira que melhore o limite superior conhecido. Desse modo, AMGI2 tende a encontrar soluções viáveis mais rapidamente, o que pode lhe permitir realizar suas podas de forma mais precoce e assim evitar a exploração desnecessária de um número maior de partições não promissoras em relação ao algoritmo de *branch-and-bound* tradicional. Ademais, essa característica de localizar soluções viáveis de forma rápida permite ainda a AMGI2 ser utilizado na qualidade de heurística de viabilidade, bastando então restringi-lo à uma determinada quantidade de tempo computacional ou número máximo de iterações, ou ainda, pará-lo quando a primeira solução viável for encontrada.

É fácil verificar que AMGI2 também apresenta convergência para uma das soluções ótimas do problema considerado com tolerância ϵ_c em um número finito de iterações, com a vantagem de não exigir otimalidade global na resolução dos problemas de minimização de *gap*. Para a escolha do próximo nó a ser explorado, pode-se escolher aquele cujo ancestral direto ficou mais próximo de zerar o *gap* antes de sua ramificação, além, é claro, dos tradicionais esquemas de busca em profundidade e largura. Por fim, pode-se escolher para a ramificação a variável y_j com valor mais distante da integralidade na solução do problema de minimização de *gap* referente à partição considerada.

4 Resultados Computacionais Preliminares

Apresentamos a seguir alguns resultados computacionais preliminares onde nossa proposta é avaliada na qualidade de heurística de viabilidade. Por hora, comparamos o desempenho de IGMA2 com as seguintes heurísticas de viabilidade para PNLIM:

- Heurística de Minimização do *Gap* de Integralidade (HMGI), de Melo et al. (2012);
- Heurística de Mergulho, versão fracionária, de Bonami and Gonçalves (2008) ;
- Feasibility Pump (FP) de Bonami and Gonçalves (2008) ;

- Feasibility Pump baseada em aproximação externa (FP-AE), versão 1, de Bonami et al. (2009a).

Utilizamos implementações próprias de AMGI2 e das demais heurísticas consideradas na linguagem C++ com compilador GCC 4.4. Para resolução dos problemas de PNL nos algoritmos, foi utilizado o pacote IPOPT 3.10 (Wächter and Biegler (2006)) sobre o pacote HSL MA27 (HSL (2007)). Para a resolução dos problemas de PLIM utilizados pelo algoritmo FP-AE foi utilizado o pacote Gurobi 4.6.1 (Gu et al. (2011)). Na implementação de HMGI e IGMA2, usamos um ponto inicial pseudo-aleatório para a resolução dos problemas de minimização de *gap* à cada iteração. Ressaltamos que a adoção dessa estratégia não faz com que nossa implementação deixe de ser determinística, pois sempre é utilizada a mesma semente de geração de números pseudo-aleatórios em cada execução, garantindo assim que a mesma sequência de números é sempre obtida.

Os testes foram rodados em um computador com processador Intel Xeon X5472 de 3,0 GHz, cache de 6 MB e memória RAM de 16 GB sob o sistema operacional Ubuntu Linux 10. Consideramos nos testes um conjunto padrão de 152 instâncias de PNLIM disponíveis em CMU-IBM (2012). Os algoritmos rodaram com tempo máximo de execução de 10 minutos para cada instância de teste. As heurísticas consideradas aqui param a execução quando a primeira solução viável é encontrada, com exceção de HMGI e AMGI2 que continuam a execução até a otimalidade ser comprovada ou o tempo máximo de execução ser atingido.

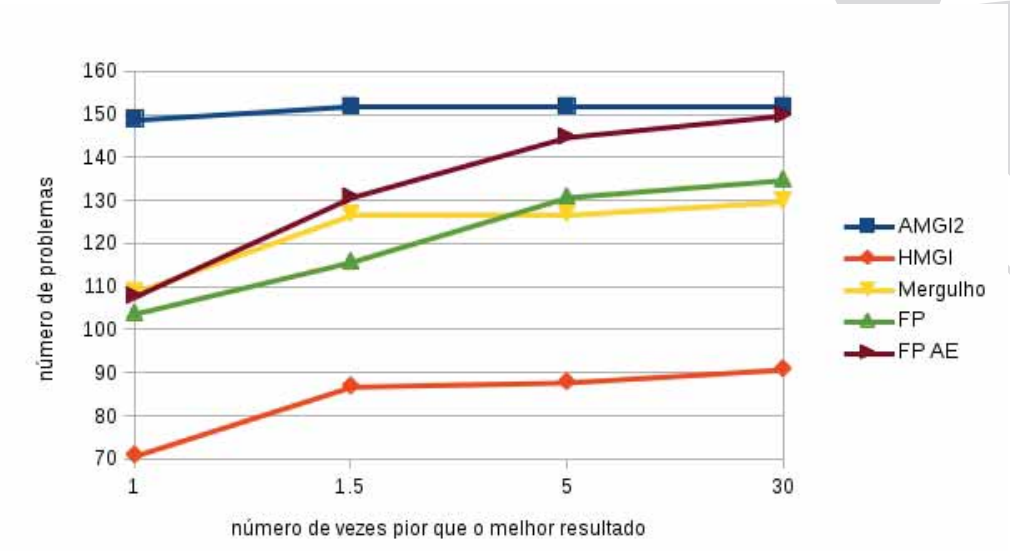

Figura 1: Comparação de performances: melhor solução encontrada.

A Figura 1, onde comparamos a qualidade das soluções encontradas, resume os resultados obtidos. A partir desta Figura, pode-se inferir que AMGI2 apresentou a melhor solução (dentre os algoritmos considerados) em 149 das 152 instâncias. É possível ainda observar que HMGI apresentou a melhor solução em apenas 71 das 152 instâncias (empates podem ocorrer). Com relação a mergulho, FP e FP-AE esse número foi de 109, 104 e 108, respectivamente. Ressaltamos que AMGI2 foi o único algoritmo a apresentar solução viável para todos os 152 problemas considerados, enquanto que para HMGI, Mergulho, FP e FP-AE esse número foi de 91, 130, 132 e 150, respectivamente. Os resultados mostram que, em pouco de tempo de execução, AMGI2 foi capaz de encontrar soluções viáveis para todas as instâncias de teste, e, em geral, essas soluções eram tão boas ou melhores do que as soluções encontradas pelas demais métodos, o que evidencia a alta competitividade do algoritmo

proposto no uso como heurística. Os resultados evidenciam ainda que a integração das resoluções dos problemas de minimização de *gap* a um procedimento de *branch-and-bound*, além de dispensar a resolução destes problemas até a otimalidade global, trouxe ainda uma considerável vantagem prática para AMGI2 em relação à HMGI. Testes computacionais futuros avaliarão AMGI2 como algoritmo exato, com uma comparação especial com o algoritmo de *branch-and-bound* não linear.

5 Conclusões

Apresentamos aqui um novo algoritmo para Programação Não Linear Inteira Mista (PNLIM) baseado na ideia de minimização de *gap* de integralidade incorporada em um procedimento de *branch-and-bound*, denominado como *Algoritmo de Minimização do Gap de Integralidade 2*, ou AMGI2. A proposta, baseada no trabalho de Melo et al. (2012) supera a principal desvantagem deste em necessitar de otimalidade global de problemas não convexos. Resolvendo problemas de minimização de *gap* de forma integrada em um único *branch-and-bound*, evitamos também a possível execução de um procedimento de *branch-and-bound* espacial diversas vezes. Como consequência, AMGI2 exige em sua implementação apenas uma rotina de Programação Não Linear local (PNL), o que de modo geral é mais simples de se obter do que uma rotina de otimização global conforme exigido em Melo et al. (2012).

AMGI2 apresenta uma relação evidente com o procedimento de *branch-and-bound* não linear padrão, com a vantagem de que, uma vez que objetiva a minimização do *gap*, tende a localizar soluções viáveis mais rapidamente, o que pode permitir podas de forma mais precoce na árvore de *branch-and-bound*, economizando assim esforço computacional. Essa característica habilita ainda o uso de AMGI2 como heurística de viabilidade e testes computacionais preliminares mostraram a competitividade do método aqui proposto nessa situação em comparação com heurísticas existentes, em especial com o trabalho de Melo et al. (2012).

Como trabalho futuro, podemos apontar a avaliação de AMGI2 como algoritmo exato para encontrar soluções ótimas de problemas de PNLIM, com especial comparação ao algoritmo de *branch-and-bound*. Um outro estudo interessante é a integração de AMGI2 na qualidade de heurística aos demais algoritmos exatos, de modo a avaliar um possível ganho de performance com solução viável obtida por AMGI2 passada a estes procedimentos. Por fim, apontamos um estudo para avaliar a utilização de diferentes algoritmos de PNL na implementação de AGMI2, uma vez que os mesmos podem fazer diferença nos resultados obtidos. Em nossos testes, utilizamos a implementação de pontos interiores do solver IPOPT, mas outros tipos de algoritmos como programação quadrática sequencial ou Lagrangeano aumentado podem se mostrar mais vantajosos. Uma implementação de AMGI2 estará livremente disponível para estudo, alteração e uso no pacote de PNLIM Muriqui, que ainda se encontra em fase de desenvolvimento.

Referências

Hsl. a collection of fortran codes for large scale scientific computation. Software, 2007.

C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier. A global optimization method, abb, for general twice-differentiable constrained {NLPs} - i. theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998. ISSN 0098-1354. doi: http://dx.doi.org/10.1016/S0098-1354(98)00027-1.

Pierre Bonami and João Gonçalves. Heuristics for convex mixed integer nonlinear programs. *Computational Optimization and Applications*, pages 1–19, 2008. ISSN 0926-6003. 10.1007/s10589-010-9350-6.

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.

Pierre Bonami, Gérard Cornuéjols, Andrea Lodi, and François Margot. A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, 119:331–352, 2009a. ISSN 0025-5610. 10.1007/s10107-008-0212-2.

Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. Algorithms and software for convex mixed integer nonlinear programs. Technical Report 1664, Computer Sciences Department, University of Wisconsin-Madison, 2009b.

Pierre Bonami, Jon Lee, Sven Leyffer, and Andreas Wächter. More branch-and-bound experiments in convex nonlinear integer programming. Technical report, 2011.

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.

Claudia D'Ambrosio and Andrea Lodi. Mixed integer nonlinear programming tools: a practical overview. *4OR*, 9(4):329–349, 2011. ISSN 1619-4500. doi: 10.1007/s10288-011-0181-9.

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.

Oliver Exler and Klaus Schittkowski. A trust region sqp algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1:269–280, 2007. ISSN 1862-4472. doi: 10.1007/s11590-006-0026-1.

Oliver Exler, Thomas Lehmann, and Klaus Schittkowski. A comparative study of sqp-type algorithms for nonlinear and nonconvex mixed-integer optimization. *Mathematical Programming Computation*, 4:383–412, 2012. ISSN 1867-2949. doi: 10.1007/s12532-012-0045-0.

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.

Ignacio E. Grossmann and Sangbum Lee. Generalized convex disjunctive programming: Nonlinear convex hull relaxation. *Computational Optimization and Applications*, 26:83–100, 2003. ISSN 0926-6003. 10.1023/A:1025154322278.

Ignacio E. Grossmann and Juan P. Ruiz. Generalized disjunctive programming: A framework for formulation and alternative algorithms for minlp optimization. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 93–115. Springer New York, 2012. ISBN 978-1-4614-1927-3. 10.1007/978-1-4614-1927-3_4.

Zonghao Gu, Edward Rothberg, and Robert Bixby. Gurobi 4.6.1. Software, December 2011.

Omprakash K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, 1985. doi: 10.1287/mnsc.31.12.1533.

Raymond Hemmecke, Matthias Köppe, Jon Lee, and Robert Weismantel. Nonlinear integer programming. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 561–618. Springer Berlin Heidelberg, 2010. ISBN 978-3-540-68279-0. 10.1007/978-3-540-68279-0_15.

Sangbum Lee and Ignacio E. Grossmann. New algorithms for nonlinear generalized disjunctive programming. *Computers & Chemical Engineering*, 24(9-10):2125 – 2141, 2000. ISSN 0098-1354. doi: 10.1016/S0098-1354(00)00581-0.

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.

Wendel Melo. Agoritmos para programação não linear inteira mista. dissertação de mestrado, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2012.

Wendel Melo, Marcia Fampa, and Fernanda Raupp. Uma heurística de minimização de gap para programação não linear inteira mista com variáveis binárias. In *XVI CLAIO/XLIV SBPO*, sep 2012.

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.

Walter Murray and Kien-Ming Ng. An algorithm for nonlinear optimization problems with binary variables. *Computational Optimization and Applications*, 47:257–288, 2010. ISSN 0926-6003. 10.1007/s10589-008-9218-1.

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.

R. Raman and I.E. Grossmann. Modelling and computational techniques for logic based integer programming. *Computers & Chemical Engineering*, 18(7):563 – 578, 1994. ISSN 0098-1354. doi: 10.1016/0098-1354(93)E0010-7. An International Journal of Computer Applications in Chemical Engineering.

Claus Still and Tapio Westerlund. Solving convex minlp optimization problems using a sequential cutting plane algorithm. *Computational Optimization and Applications*, 34:63–83, 2006. ISSN 0926-6003. 10.1007/s10589-005-3076-x.

Robert A. Stubbs and Sanjay Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86:515–532, 1999. ISSN 0025-5610. 10.1007/s101070050103.

Francisco Trespalacios and Ignacio E. Grossmann. Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86(7):991–1012, 2014. ISSN 1522-2640. doi: 10.1002/cite.201400037.

Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006. ISSN 0025-5610. 10.1007/s10107-004-0559-y.

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.