NOVAS ABORDAGENS DE SOLUÇÃO PARA PROGRAMAÇÃO NÃO LINEAR INTEIRA MISTA BINÁRIA E PROGRAMAÇÃO QUADRÁTICA NÃO CONVEXA.

Wendel Alexandre Xavier de Melo

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Marcia Helena Costa Fampa
Fernanda Maria Pereira Raupp
Jon Lee

Rio de Janeiro
Março de 2016

NOVAS ABORDAGENS DE SOLUÇÃO PARA PROGRAMAÇÃO NÃO LINEAR INTEIRA MISTA BINÁRIA E PROGRAMAÇÃO QUADRÁTICA NÃO CONVEXA.

Wendel Alexandre Xavier de Melo

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Profª. Marcia Helena Costa Fampa, D.Sc.

Profª. Fernanda Maria Pereira Raupp, D.Sc.

Prof. Jon Lee, Ph.D.

Prof. Abilio Pereira de Lucena Filho, D.Sc.

Prof. Eduardo Uchoa Barboza, D.Sc.

Prof. Luidi Gelabert Simonetti, D.Sc.

Prof. Marcus Vinicius Soledade Poggi de Aragao, Ph.D.

Profª. Sandra Augusta Santos, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2016

Melo, Wendel Alexandre Xavier de

Novas abordagens de solução para programação não linear inteira mista binária e programação quadrática não convexa./Wendel Alexandre Xavier de Melo. – Rio de Janeiro: UFRJ/COPPE, 2016.

XVI, 102 p.: il.; 29,7cm.

Orientadores: Marcia Helena Costa Fampa
Fernanda Maria Pereira Raupp
Jon Lee

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 82 – 90.

1. Programação Não Linear Inteira Mista. 2. Programação Quadrática Não Convexa. 3. Otimização Global. 4. *Branch-and-Bound*. 5. *Branch-and-Bound* Espacial. 6. Heurísticas. 7. Minimização de Gap. 8. Computação Científica. 9. Desenvolvimento de *Solvers*. I. Fampa, Marcia Helena Costa *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Às minhas avós Lenir e Vanda,
à (tia) Deise Lobo Cavalcante
(in memoriam) e às minhas
"mães acadêmicas" Marcia
Fampa e Fernanda Raupp.

Agradecimentos

Agradeço a Deus por, apesar de todos os entraves e dificuldades, ter me dado a oportunidade de chegar até aqui. Agradeço às minhas avós, Lenir e Vanda, pelo apoio e por desde sempre terem dado suporte aos meus estudos. Agradeço ainda à toda a minha família pela confiança em mim, em especial aos meus pais, que, embora não compreendam bem o meu trabalho, o respeitam e se mostram orgulhosos por mim.

Agradeço às minhas queridas orientadoras, Marcia Fampa e Fernanda Raupp por toda a compreensão, incentivo, apoio, descontração e ensinamentos ao longo de todos esses anos em que elas têm me aturado. Sem dúvida, o maior prêmio que eu poderia receber na minha vida acadêmica é ter duas orientadoras como elas, que sempre me deram autonomia e confiança além de me deixarem muito confortável para desenvolver meus trabalhos de pesquisa. Pude testemunhar situações onde alunos não se davam bem com seus orientadores e vi o quanto isto pode ser ruim. Sou grato a Deus por ter colocado esses dois anjos na minha vida que são mais do que orientadoras para mim, são verdadeiras conselheiras, amigas, confidentes, enfim, minhas "mães acadêmicas" por quem sempre terei todo o respeito do mundo.

Pouca gente sabe, mas eu tive uma terceira mãe acadêmica, a minha orientadora no ensino médio, professora Maria Amélia Costa, que muito me ensinou sobre o método de trabalho científico e que despertou a minha vocação para a ciência. Essa tese não poderia passar sem um sincero agradecimento a ela, e eu aqui o faço. Aproveito também para agradecer a toda equipe de pessoas da minha saudosa Escola Politécnica de Saúde Joaquim Venâncio da Fiocruz, onde pude conhecer pessoas especiais, fazer amigos para toda a vida, onde aprendi a ser gente, onde pude me encontrar, me reconhecer, adquirir opinião e desenvolver habilidades como nunca antes na minha vida. Sem dúvida, o mundo seria um lugar muito melhor se todos pudessem obter a educação de qualidade ímpar que eu obtive naquele lugar encantado. Ainda nesse contexto, agradeço a todos os meus professores que atuaram com paixão e empenho nos mais diversos níveis da minha formação.

Parte do meu trabalho de pesquisa foi desenvolvida enquanto eu estive visitando a Universidade de Michigan (UMich), Ann Arbor, EUA. Agradeço ao Departamento de Engenharia Industrial e Operações desta universidade por me aceitar como pes-

quisador visitante, em especial ao professor Jon Lee que aceitou me receber e orientar por lá, apesar de todas as minhas dificuldades com inglês, e ainda me ajudou provendo parte do meu suporte financeiro. Sem dúvida o período em que passei em Ann Arbor foi bastante enriquecedor na minha formação pessoal e profissional. Sou grato também às pessoas especiais que passaram pela minha vida enquanto estive nos EUA: Rebecca Pagels, minha melhor amiga, que sempre teve hiper paciência e se importou comigo de forma sincera, sem esperar nada em troca; *Lady* Maria Alcini, por todos os momentos especiais; Marina Lualdi, a minha estimada professora de yoga e nia; Tina Picano, meu anjo na universidade, a secretária do departamento que sempre tinha dicas para me ajudar a resolver os meus problemas acadêmicos e na vida americana de um modo geral; Chris Konrad, funcionário de departamento que me ajudava com o *hardware*; Nancy, a auxiliar de serviços gerais sempre tão simpática com quem podia praticar inglês e espanhol; Gwendolyn Brown e Candy, que também eram muito legais comigo na UMich. Agradeço ainda à brasileira Arlete Silva por me acolher e me ensinar importantes lições sobre humildade.

A saudade de casa, da família e dos amigos, bem como as pressões do trabalho teriam me enlouquecido (ainda mais) em Ann Arbor se não fosse por um lugar mágico que conheci naquela cidadezinha pitoresca. Um lugar onde eu pude ser eu mesmo, onde eu pude sorrir sem me importar com mais nada, onde pude me encontrar com a minha alegria interna e me expressar sem qualquer censura. Agradeço à toda gente do *Dance Revolution*, a melhor escola de dança do mundo, que, não por acaso, possui a melhor professora de Salsa do mundo, Laura Geldys, a quem sou muito grato por ter me proporcionando a inenarrável experiência de aprender um pouco da magnífica Salsa Nova Iorque. Tive a oportunidade de estar com pessoas muito especiais naquele salão de dança que para mim chega a ser quase sagrado, e, agradeço de coração a todos! Agradeço também ao povo do grupo de Salsa da UMich, o *MSalsa*, onde pude aprender um pouco da tradicional Salsa cubana conhecida como *Rueda de Casino*. Ainda nesse gancho, agradeço aos amigos bailadores Ashley Duan, Ed Trager, Kat Curtis, Savi Sachdev, Carly Faver, Janani Viswanathan, Gerardo Cruz, Leila Jam, TingYing Lee, Cindy, Aisha, Karen Wilkerson, Tre, Jessica, Sabrina, Estefanía Perez, Liza Krylova, Carolina Simao, Omar Saadeh, Maria Elena, Annie Klink, além de muitos outros que não consigo enumerar aqui! *¡Azúcar a todos mis compañeros del Casino y a los de Nueva Iorque!*.

Agradeço aos funcionários do PESC que sempre me socorrem quando preciso de ajuda no programa de pós-graduação: Solange, Claudia, Sonia, Mercedes, Fátima, Gutierrez e Roberto Rodrigues. À Deise Lobo Cavalcanti (*in memoriam*), a minha singela homenagem por toda sua dedicação a frente da secretaria do Instituto de Matemática da UFRJ. Certamente, a perda da Tia Deise é fato irreparável para todos nós, e estou certo de que a universidade jamais terá outra funcionária tão

dedicada, fonte de inspiração à todas as pessoas bem intencionadas e de bom coração. Que ela esteja descansando em paz e olhando por todos nós. Agradeço também aos meus queridos alunos na universidade, do período em que fui professor substituto do Departamento de Ciência da Computação, alunos estes que sempre tiveram grande respeito e carinho por mim.

Teria sido mais difícil chegar até aqui se não fosse a valiosíssima ajuda que alguns companheiros de faculdade me forneceram ao me doar um computador, onde pude dar os primeiros passos como pesquisador em meus trabalhos de iniciação científica. Um grande abraço para Vitor Gamboa, Carlos Eduardo Gomes (KDU), Miguel Gabriel Carvalho, Antônio Henrique (Montanha) e Thiago Loureiro (Escazi).

Agradeço ainda à minha amiga Priscilla Luz pela preciosa amizade ao longo desses últimos anos.

Agradeço aos desenvolvedores de *softwares* livres que trabalham incansavelmente para nos disponibilizar excelentes ferramentas, algumas das quais me serviram de base para desenvolver o meu trabalho. Em particular, um agradecimento especial aos desenvolvedores dos aplicativos `GCC`, `Valgrind`, `Kdevelop`, `GLPK`, `Ipopt`, `Csdp`, `Octave`, `LAPACK`, `BLAS` e da plataforma `Linux` de um modo geral. Agradeço também às empresas e organizações que me forneceram licenças para uso de seus produtos comerciais: `Mosek`, `IBM` (`Cplex`), `Gurobi`, `Intel` (`icc/mkl`), `GAMS`, `Optimization Firm` (`Baron`), `ESA` (`Worhp`), `Pardiso` e `Research Councils UK (HSL)`.

Agradeço aos membros da banca por aceitarem o convite para avaliar este trabalho de pesquisa.

Por fim, agradeço a todos os que trabalharam, e os que ainda trabalham, de forma assídua e ética pelo desenvolvimento do país, da educação e da ciência brasileira, e que, deste modo, abriram-me o mcaminho para que eu pudesse chegar até aqui. Eu tenho a mais profunda esperança de, com meu trabalho, também dar a minha contribuição pelo avanço da sociedade, do país e do mundo, e ainda incentivar outras pessoas a também trabalhare em prol dessa mesma finalidade.

> "*A utopia está lá no horizonte. Me aproximo dois passos, ela se afasta dois passos. Caminho dez passos e o horizonte corre dez passos. Por mais que eu caminhe, jamais alcançarei. Para que serve a utopia? Serve para isso: para que eu não deixe de caminhar.*"
>
> (Eduardo Galeano, a partir de uma resposta dada por Fernando Birri sobre a utilidade da utopia)

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

NOVAS ABORDAGENS DE SOLUÇÃO PARA PROGRAMAÇÃO NÃO LINEAR INTEIRA MISTA BINÁRIA E PROGRAMAÇÃO QUADRÁTICA NÃO CONVEXA.

Wendel Alexandre Xavier de Melo

Março/2016

Orientadores: Marcia Helena Costa Fampa
Fernanda Maria Pereira Raupp
Jon Lee

Programa: Engenharia de Sistemas e Computação

Este trabalho é fundamentalmente dividido em duas partes. Na primeira parte, elaboramos uma nova metodologia para a abordagem do problema convexo de Programação Não Linear Inteira Mista (PNLIM) binária. Apresentamos aqui duas versões de um algoritmo exato para solucionar o problema mencionado baseadas na ideia de minimização do *gap* de integralidade. Demonstramos aqui a convergência dos algoritmos em número finito de iterações. As abordagens propostas podem localizar soluções viáveis rapidamente, habilitando assim seu uso na qualidade de heurística de viabilidade. Na segunda parte, tratamos do problema de otimização global onde os termos não convexos se remetem a expressões quadráticas na função objetivo e/ou restrições. Desenvolvemos uma abordagem baseada em decomposição por Diferença de duas funções Convexas (DC) para a construção de relaxações convexas adotadas em um procedimento de *Branch-and-Bound* Espacial (BBE). Apresentamos diferentes estratégias de decomposição e demonstramos a equivalência de algumas das mesmas. Em ambas as partes, fornecemos resultados computacionais promissores. Complementarmente às contribuições teóricas, desenvolvemos contribuições práticas por meio de pacotes computacionais que implementam as abordagens aqui descritas. Estes pacotes computacionais serão disponibilizados à comunidade técnica e científica para uso livre.

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

NEW SOLUTION APPROACHES TO BINARY MIXED INTEGER NONLINEAR PROGRAMMING AND NON-CONVEX QUADRATIC PROGRAMMING.

Wendel Alexandre Xavier de Melo

March/2016

Advisors: Marcia Helena Costa Fampa
Fernanda Maria Pereira Raupp
Jon Lee

Department: Systems Engineering and Computer Science

This thesis is divided into two parts. In the first one, we develop a new methodology to treat binary Mixed Integer NonLinear Programming (MINLP) problems. We present two versions of an exact algorithm to solve the addressed problem based on integrality gap minimization. We demonstrate the convergence of the algorithms in a finite number of iterations. The proposed approaches can find feasible solutions quickly, enabling its usage as a feasibility heuristic. In the second part, we deal with the global optimization problem where noncmonvex terms only appear in quadratic expressions in the objective function and/or constraints. We develop an approach based on the Difference of two Convex functions (DC) in order to build a convex relaxation to be used in a Spatial Branch-And-Bound procedure. We present different decomposition strategies and we prove the equivalence of some of the. In both parts, we provide promising computational results. We also develop practical contributions by means of computational packages that implement the approaches described here. Those packages will be available to technicians and researchers for free usage.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AMGI1	Algoritmo de Minimização do Gap de Integralidade 1, p. 5
AMGI2	Algoritmo de Minimização do Gap de Integralidade 2, p. 6
BBE	*Branch-And-Bound* Espacial, p. 15, 44
BB	Branch-And-Bound, p. 3
DC	Diferença de (funções) convexas, p. 40
PLIM	Programação Linear Inteira Mista, p. 3
PNLIM	Programação Não Linear Inteira Mista, p. 2
PNL	Programação Não Linear, p. 3, 37

Capítulo 1

Apresentação Geral

Este trabalho é essencialmente dividido em duas partes distintas. Em cada uma dessas partes, é tratado um problema de otimização específico, para o qual desenvolvemos abordagens de solução e apresentamos resultados computacionais consistentes.

Na Parte I, abordamos o problema convexo de Programação Não Linear Inteira Mista (PNLIM) binária:

$$\begin{array}{lll} (P) & \min\limits_{x,\,y} & f(x,y) \\ & \text{sujeito a:} & g(x,y) \leq 0, \\ & & x \in X,\ y \in Y \cap \{0,1\}^{n_y}, \end{array} \tag{1.1}$$

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. Foram desenvolvidos dois algoritmos para solucionar o problema (P) de forma exata baseados na ideia de minimização de *gap* de integralidade. Por essa razão, os mesmos foram denominados como Algoritmos de Minimização de *Gap* de Integralidade (AMGI). Ambas as versões de AMGI são apresentadas em detalhes e a convergência para solução ótima em número finito de iterações é demonstrada. As abordagens AMGI possuem a peculiaridade de poderem localizar soluções viáveis de forma bastante rápida, o que habilita seu uso como heurística de viabilidade, podendo ter, nesse contexto, sua execução interrompida de forma precoce, isto é, antes da otimalidade ser comprovada. Realizamos experimentos computacionais avaliando o desempenho de AMGI em relação a outros algoritmos exatos de PNLIM encontrados na literatura. Adicionalmente, também comparamos AMGI com alguns algoritmos heurísticos. Em ambos os casos, conseguimos resultados promissores.

Na Parte II, tratamos o problema não convexo de Programação Não Linear onde a não convexidade se remete apenas a termos quadráticos existentes na função

objetivo e/ou restrições:

$$
\begin{array}{lll}
(PQR) \quad z := \min\limits_{x} & f_0(x) \ + \ q_0(x), & \\
\text{sujeito a:} & f_i(x) \ + \ q_i(x) \leq b_i, \quad i = 1, \ldots, m & \\
& l_x \leq x \leq u_x, & \\
& x_j \in \mathbb{Z}, \text{ para } j \in \mathcal{I}, &
\end{array}
\tag{1.2}
$$

onde, para $j = 0, \ldots, m$, $f_j(x)$ são funções não lineares convexas e $q_j(x) = \frac{1}{2} x' Q_j x$ são funções quadráticas. Assumimos aqui que a região viável de (PQR) é limitada e que ao menos alguma das matrizes simétricas $Q_j, \ j = 0, \ldots, m$ não é semidefinida positiva, o que significa dizer que, ao menos uma das funções $q_j(x)$ é não convexa. Observe que o problema (PQR) pode incorporar variáveis em domínios inteiros, o que o torna um problema de PNLIM não convexa. Uma abordagem baseada em *Branch-And-Bound* Espacial (BBE) é proposta para o problema (PQR). Apresentamos diferentes estratégias de decomposição da parte não convexa de (PQR) com o propósito de construir uma relaxação convexa funcional para este problema a ser adotada em nosso algoritmo baseado em BBE. Novamente, são apresentados os resultados computacionais onde avaliamos o desempenho de diversas estratégias de decomposição. Adicionalmente, comparamos nossos melhores resultados com os de conhecidos pacotes computacionais comumente utilizados para solucionar o problema (PQR). Nesse último contexto, o trabalho aqui desenvolvido se mostrou extremamente competitivo.

Ambas as partes tiveram como objetivo adicional, além da contribuição teórica por meio da pesquisa sobre algoritmos, a apresentação de uma contribuição prática computacional por meio da construção de *solvers* que implementam as abordagens aqui propostas. Os pacotes computacionais aqui desenvolvidos, descritos brevemente ao final de cada parte, serão disponibilizados à comunidade técnica e científica como *software* aberto. Esperamos desse modo contribuir de forma direta e efetiva para o desenvolvimento não apenas das áreas aqui estudadas, mas do conhecimento científico como um todo nos seus mais diversos campos de atuação.

Parte I

Algoritmos de Minimização de *Gap* para Programação Não Linear Inteira Mista Binária

Capítulo 2

Introdução

2.1 Definição do problema

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

$$\begin{array}{lll} (P) \quad \min\limits_{x,\,y} & f(x,y) & \\ \text{sujeito a:} & g(x,y) \leq 0, & \\ & x \in X,\ y \in Y \cap \{0,\,1\}^{n_y}, & \end{array} \tag{2.1}$$

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. Uma vez que as funções de (P) são convexas, o problema (P) é dito convexo, embora, a simples presença de variáveis binárias o torne não convexo a rigor. Destacamos que uma variável inteira não binária com limitantes inferior e superior finitos pode ser substituída por uma expressão linear em função apenas de variáveis binárias. Deste modo, mesmo um problema de PNLIM não binária pode ser resolvido por abordagens para PNLIM binária, embora, na prática, esse tipo de transformação não seja recomendada por aumentar a dificuldade envolvida na resolução do problema.

PNLIM tem sido objeto de estudo de diversos pesquisadores em tempos recentes. O leitor interessado pode conferir bons trabalhos de revisão bibliográfica em [1–5]. Dentre as aplicações de PNLIM encontradas na literatura, podemos mencionar: planejamento de redes de telecomunicações [6], agendamento de operações em refinarias de petróleo [7], sistemas de geração de energia [8], projetos de cadeias de

suprimentos [9], caminhos de robôs industriais [10], síntese de processos químicos [11], prevenção de conflitos de aeronaves [12], redes metabólicas em biotecnologia [13], redes de distribuição de água [14], trajetória de válvulas em séries de tanques em cascatas [15], projeto de camada de isolante térmico para o Grande Colisor de Hádrons [16] e recarga de núcleo de reatores nucleares [16].

2.2 Classes de algoritmos para PNLIM

Ao longo dos últimos anos, diversos algoritmos foram apresentados para a resolução de problemas de PNLIM. Neste trabalho, formulamos uma classificação própria para enquadrar os algoritmos existentes nas seguintes categorias:

1. **Algoritmos de aproximação linear:** Este tipo de algoritmo 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 abordado. A principal vantagem destes algoritmos é que eles se valem diretamente de todo o avanço e maturidade já alcançados na área de PLIM, o que tem se mostrado especialmente eficiente em casos onde o grau de não linearidade do problema de PNLIM abordado é baixo. O principal ponto fraco é que, 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, caso muitas linearizações sejam necessárias até que o problema original esteja adequadamente aproximado. Os principais representantes dessa classe de algoritmos são os algoritmos de decomposição de Benders generalizada [17], aproximação externa [11, 18], *branch-and-bound* linear/não linear baseado em aproximação externa [19] e plano de corte estendido [20].

2. **Algoritmos de *branch-and-bound* não linear:** De modo geral, esta classe generaliza 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 auxiliares para aceleração de sua convergência tais como pré-processamento, (re)início promissor (*hot start*), *strong branching*, e geração de cortes. Estas técnicas em geral não são facilmente estendidas ao universo não linear, o que, aliado a uma relativa 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 [21–26].

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 [27, 28], embora, segundo nosso melhor conhecimento, não exista ainda demonstração de convergência para estes algoritmos. Temos ainda em [29] um algoritmo baseado em lagrangeano para problemas separáveis ou polinomiais 0-1 e o trabalho apresentado em [30], 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.

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. Vale destacar que, neste caso, a recíproca se mostra verdadeira, isto é, problemas de Programação Disjuntiva podem ser reformulados e resolvidos como problemas de PNLIM. O leitor interessado no assunto pode consultar os recentes trabalhos [2, 31–34].

5. **Abordagens híbridas:** Os algoritmos dessa categoria procuram combinar características presentes nos algoritmos das categorias anteriores. O objetivo principal é aproveitar as principais vantagens das abordagens mescladas e, de forma simultânea, remediar seus pontos fracos. Como representantes dessa classe, podemos mencionar o algoritmo em [35], que mistura o algoritmo de BB linear/não linear baseado em aproximação externa com o algoritmo de aproximação externa puro, e [36], que mistura *branch-and-bound* não linear padrão com aproximação externa.

6. **Heurísticas:** Em geral, abordagens dessa classe têm como objetivo fornecer soluções viáveis para os problemas abordados (heurísticas de viabilidade), ou melhorar soluções viáveis já obtidas para os mesmos (heurísticas de melhoramento). Heurísticas podem utilizar mecanismos dos mais variados tipos para cumprir suas metas, inclusive apresentar características de algoritmos

das classes descritas anteriormente. Todavia, preferimos separar as heurísticas em uma categoria a parte devido ao seu não compromisso com a otimalidade do problema abordado. Em outras palavras, a aplicação de uma heurística não traz a garantia da obtenção de uma solução ótima, mas, ainda assim, esses algoritmos têm suma importância pelo fato de poderem ser utilizados em conjunto com os demais algoritmos para acelerar a convergência destes. Uma boa solução viável, juntamente com o limitante superior (minimização) que a mesma fornece, pode ser de grande utilidade para:

(a) Melhorar aproximações lineares;

(b) Evitar a exploração desnecessária de sub-ramos da árvore de enumeração dos algoritmos de BB, ou ao menos diminuir a demanda por memória destes uma vez que, um bom limitante superior pode diminuir o tamanho máximo da lista de sub-ramos em aberto;

(c) Ser utilizada como um bom ponto de partida para os algoritmos de reformulação contínua.

Ademais, em alguns casos, obter uma solução viável para um problema de PNLIM pode se constituir por si só em uma tarefa bastante árdua, sendo até mesmo impraticável para alguns algoritmos não heurísticos cumpri-la em tempo razoável. O desenvolvimento e aplicação de heurísticas para PNLIM são relatados em [1, 37–41].

Ressaltamos que, dentre outras razões, por ser esta uma área de estudo relativamente recente, ainda não existe uma abordagem com desempenho prático claramente superior às demais. Além disso, os diversos algoritmos até então apresentados muitas vezes apresentam performance prática que deixa a desejar, demandando grande esforço computacional para a resolução de problemas de tamanho modesto em comparação aos problemas já resolvidos pelos algoritmos para PLIM. Estes fatos justificam plenamente as pesquisas pela melhoria dos algoritmos já existentes, bem como aquelas por novas abordagens promissoras para PNLIM, tanto em âmbito de metodologias exatas quanto de metodologias heurísticas.

2.3 Escopo

Esta parte do trabalho inicialmente propõe, no Capítulo 3, um algoritmo da classe de subproblemas contínuos para a resolução de (P) denominado como *Algoritmo de Minimização do Gap de Integralidade 1* ou AMGI1 . AMGI1 é baseado na ideia da resolução de problemas não convexos visando a minimizar o *gap* de integralidade e, assim, encontrar uma sequência de soluções viáveis até a solução ótima ser

encontrada. Posteriormente, no Capítulo 4, desenvolvemos uma segunda versão do algoritmo denominada como AMGI2, onde nos aproveitamos da mesma ideia da versão anterior incorporada em procedimento de *branch-and-bound*. Embora AMGI1 e AMGI2 tenham sido formulados para a resolução de (P) até a sua otimalidade, percebemos que estes algoritmos tendem a localizar soluções viáveis rapidamente, o que nos motivou a apresentar uma pequena modificação em AMGI2 para tornar mais efetivo o seu uso como heurística de viabilidade no Capítulo 4. Resultados computacionais são apresentados no Capítulo 5, onde desenhamos ainda algumas conclusões. Adicionalmente, fazemos proveito deste capítulo para apresentar uma das maiores contribuições produzidas por este trabalho de pesquisa, que consiste no oferecimento à comunidade técnica e científica de um novo pacote computacional aberto para PNLIM, o *solver* `Muriqui`.

Capítulo 3

O Algoritmo de Minimização do *Gap* de Integralidade (versão 1)

3.1 O problema abordado

Conforme mencionado no Capítulo 2, nesta parte do trabalho abordamos o seguinte problema convexo de Programação Não Linear Inteira Mista (PNLIM) onde todas as variáveis inteiras são binárias:

$$\begin{array}{lll} (P) & \min\limits_{x,\,y} & f(x,y) \\ & \text{sujeito a:} & g(x,y) \leq 0, \\ & & x \in X,\ y \in Y \cap \{0,1\}^{n_y}. \end{array} \tag{3.1}$$

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 e duplamente diferenciáveis e contínuas. Assumimos aqui que (P) possui solução ótima.

3.2 Subproblemas Considerados

Algoritmos que tratam o problema (P) comumente abordam a versão relaxada deste na qual as restrições de integralidade são omitidas. Desse modo, constrói-se o seguinte problema de Programação Não Linear contínua (PNL), denominado relaxação contínua de (P):

$$\begin{array}{lll} (\tilde{P}) & \min\limits_{x,\,y} & f(x,y) \\ & \text{sujeito a:} & g(x,y) \leq 0, \\ & & 0 \leq y \leq 1, \\ & & x \in X,\ y \in Y. \end{array} \tag{3.2}$$

Seja $(\tilde{x}, \tilde{y})$ uma solução qualquer para $(\tilde{P})$. Definimos o *gap* de integralidade

(ou simplesmente *gap*) de $(\tilde{x}, \tilde{y})$ como sendo uma medida para a distância que esta solução está de satisfazer as restrições de integralidade relaxadas em $(\tilde{P})$. Uma das possíveis formas de se calcular este *gap* de integralidade é através da expressão:

$$\text{gap}(y) = \sum_{i=1}^{n_y} y_i(1 - y_i)$$

Observe que, obviamente, o *gap* não depende das variáveis contínuas $\tilde{x}$. Note ainda que, considerar as restrições de integralidade de (P) consiste em considerar na região viável deste problema apenas as soluções onde o respectivo *gap* é zerado. Deste modo, nossa motivação para o desenvolvimento de um novo algoritmo para PNLIM binária vem do fato de que podemos substituir as restrições de integralidade pelas restrições $\text{gap}(y) = 0$ e $0 \leq y \leq 1$, obtendo assim o seguinte problema de PNL:

$$\begin{aligned}
(\bar{P}) \qquad \min_{x,\,y} \quad & f(x, y) && (3.3)\\
\text{sujeito a:} \quad & g(x, y) \;\leq\; 0, && (3.4)\\
& \sum_{i=1}^{n_y} y_i(1 - y_i) \;=\; 0, && (3.5)\\
& 0 \leq y \leq 1, && (3.6)\\
& x \in X,\; y \in Y. && (3.7)
\end{aligned}$$

Embora $(\bar{P})$ seja um problema apenas em domínio contínuo, o que em princípio simplificaria seu processo de resolução, a restrição (3.5) o torna não convexo e com região viável desconexa, o que dificulta de forma bastante acentuada o processo prático de obtenção de uma de suas soluções ótimas globais. Por esta razão, Murray e Ng propõem em [30] um algoritmo para a resolução de um problema de PNLIM com variáveis binárias e restrições lineares onde a restrição (3.5) é penalizada na função objetivo. Para lidar com a não-convexidade da nova função objetivo, uma estratégia de suavização é adotada pelos autores. Este mesmo recurso de penalizar a restrição (3.5) na função objetivo já havia sido anteriormente abordado por Raghavachari em [42], ainda no âmbito de Programação Linear Inteira Mista (PLIM).

Diante das dificuldades trazidas pela incorporação da restrição (3.5), e notando que o *gap* de uma solução viável de $\tilde{P}$ tem como menor valor possível o valor zero, consideramos a resolução de outro problema de otimização onde o objetivo consiste em apenas minimizar o *gap* de integralidade:

$$
\begin{aligned}
\left(P^G\right) \qquad \min_{x,\,y} \quad & \sum_{i=1}^{n_y} y_i(1-y_i) && (3.8)\\
\text{sujeito a:} \quad & g(x,y) \;\leq\; 0, && (3.9)\\
& 0 \leq y \leq 1, && (3.10)\\
& x \in X,\; y \in Y. && (3.11)
\end{aligned}
$$

Denominamos o problema (P^G) como *problema de minimização de gap de integralidade*. Embora a função objetivo de (P^G) seja côncava (não convexa), o que ainda traz dificuldades para a sua resolução, sua região viável é convexa, o que favorece o processo de obtenção de soluções por parte dos pacotes de PNL em geral. Observe ainda que este problema não considera a função objetivo original de (P), tendo como único foco a obtenção de uma solução viável para o mesmo. É facilmente verificável que se a região viável de (P) é não vazia, uma solução $(\bar{x},\bar{y})$ é ótima global de (P^G) se, e somente se, a mesma é viável para (P).

A obtenção de uma solução ótima global para o problema (P^G) nos dá uma solução viável para (P). Entretanto, nada podemos aferir sobre a qualidade dessa solução, já que a função objetivo original de (P) não é considerada nesse novo problema. Sendo assim, qualquer solução viável para (P) pode ser obtida com este processo, inclusive alguma que maximize a função objetivo $f(x,y)$ no lugar de alguma que a minimize. Para remediar este fato, adotamos duas estratégias visando a melhoria da solução encontrada. A primeira estratégia consiste em realizar uma espécie de busca local sobre a vizinhança dessa solução. Tomando por $(\bar{x},\bar{y})$ a solução ótima global obtida para (P^G), realizamos uma busca local em sua vizinhança resolvendo o seguinte problema de PNL obtido de (P) ao se fixar y em $\bar{y}$:

$$
\begin{aligned}
(P(\bar{y})) \quad \min_{x} \quad & f(x,\bar{y}) \\
\text{sujeito a:} \quad & g(x,\bar{y}) \;\leq\; 0, \\
& x \in X.
\end{aligned}
\qquad (3.12)
$$

Observe 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 (no caso de $(P(\bar{y}))$ possuir múltiplas soluções ótimas, $(\tilde{x},\bar{y})$ será uma das melhores soluções para (P) com y fixo em $\bar{y}$). É importante mencionar que esta estratégia de resolver o problema $(P(\bar{y}))$ é adotada em certos algoritmos para PNLIM que se valem de aproximação linear - como decomposição de Benders generalizada [17] e aproximação externa [11] - para obtenção de limitantes superiores válidos para (P) e fortalecimento, por meio da solução obtida, das respectivas relaxações de PLIM utilizadas por estes algoritmos.

Nossa segunda estratégia de melhoria de solução consiste na resolução do seguinte problema PNL, que também visa minimizar o *gap* de integralidade e adota uma restrição de corte de nível objetivo:

$$
\begin{aligned}
(\hat{P}^G(z^u, \epsilon_c)) \qquad \min_{x,\,y} \quad & \sum_{i=1}^{n_y} y_i(1 - y_i) && (3.13)\\
\text{sujeito a:} \quad & g(x, y) \;\leq\; 0, && (3.14)\\
& f(x, y) \;\leq\; z^u - \epsilon_c, && (3.15)\\
& 0 \leq y \leq 1, && (3.16)\\
& x \in X,\; y \in Y. && (3.17)
\end{aligned}
$$

onde z^u é o melhor limitante superior obtido para (P), e $\epsilon_c > 0$ é uma tolerância de convergência. Denominamos o problema $(\hat{P}^G(z^u, \epsilon_c))$ como *problema de minimização de gap de integralidade com corte de nível objetivo*. Note que a única diferença entre os problemas (P^G) e $(\hat{P}^G(z^u, \epsilon_c))$ é que este último traz a restrição de corte de nível objetivo (3.15), que visa a obtenção de uma solução viável que melhore o limitante superior conhecido, quando possível.

3.3 Algoritmo e Demonstração de Convergência

Tendo apresentado os subproblemas de interesse, estamos prontos para introduzir a primeira versão de nosso algoritmo para solucionar o problema (P), denominado aqui como *Algoritmo de Minimização de Gap de Integralidade*. Designamos esta primeira versão do algoritmo como AMGI1 e o apresentamos como Algoritmo 3.1. Observe que uma solução $(\bar{x}^0, \bar{y}^0)$ viável para (P) pode ser obtida na linha 1 com a resolução de (P^G). Em caso de sucesso, usamos o valor $\bar{y}^0$ para a resolução de $(P(\bar{y}^0))$ na expectativa de melhorar a solução obtida. Observe que o laço entre as linhas 7 e 15 considera o problema $(\hat{P}^G(z^u, \epsilon_c))$ no lugar de (P^G) e tem como objetivo obter uma sequência de soluções $(\tilde{x}^k, \bar{y}^k)$ cada vez melhores com respeito à função objetivo, até que $(\hat{P}^G(z^u, \epsilon_c))$ seja inviável ou sua resolução forneça uma solução $\bar{y}^k$ não inteira. O teorema a seguir assegura a convergência do algoritmo AMGI1 em um número finito de iterações se otimalidade global puder ser assegurada na resolução de (P^G) e $(\hat{P}^G(z^u, \epsilon_c))$.

Teorema 3.1. *Sejam (P^G) e $(\hat{P}^G(z^u, \epsilon_c))$ os problemas de minimização de gap de integralidade tomados pelo algoritmo AMGI1 (Algoritmo 3.1). 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 ϵ_c, em um número finito de iterações.*

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

1 Seja $(\bar{x}^0, \bar{y}^0)$ uma solução ótima global de (P^G) ;
2 **se** $\bar{y}^0$ *for inteira* **então**
3 Seja $\tilde{x}^0$ uma solução ótima de $(P(\bar{y}^0))$;
4 $(x^*, y^*) = (\tilde{x}^0, \bar{y}^0)$;
5 $z^u = f(\tilde{x}^0, \bar{y}^0)$;
6 $k = 1$;
7 **enquanto** $(\hat{P}^G(z^u, \epsilon_c))$ *for viável* **faça**
8 Seja $(\bar{x}^k, \bar{y}^k)$ uma solução ótima global de $(\hat{P}^G(z^u, \epsilon_c))$;
9 **se** $\bar{y}^k$ *for inteira* **então**
10 Seja $\tilde{x}^k$ uma solução ótima de $(P(\bar{y}^k))$;
11 $(x^*, y^*) = (\tilde{x}^k, \bar{y}^k)$;
12 $z^u = f(\tilde{x}^k, \bar{y}^k)$;
13 **senão**
14 **pare o algoritmo** ;
15 $k = k + 1$;

Algoritmo 3.1: Algoritmo AMGI1.

Demonstração. Uma vez que o problema (P^G) é resolvido até a otimalidade global, se (P) for viável, o algoritmo AMGI1 obterá, na linha 1, uma solução viável inicial $(\bar{x}^0, \bar{y}^0)$ com $\bar{y}^0$ inteira. A resolução de $(P(\bar{y}^0))$ nos dará a solução $(\tilde{x}^0, \bar{y}^0)$, que, então, será a melhor solução viável de (P) tendo $\bar{y}^0$ como valor para y e o valor $f(\tilde{x}^0, \bar{y}^0)$ será utilizado como valor inicial para o limitante superior z^u.

A partir de então, o algoritmo considera o problema $(\hat{P}^G(z^u, \epsilon_c))$, que possui restrição de corte de nível objetivo sobre o valor corrente de z^u. Uma vez que esse corte é feito primeiramente sobre o valor da melhor solução que possui $\bar{y}^0$ como valor para y, podemos garantir que $\bar{y}^0$ não pode figurar como valor de y em nenhuma solução viável de $(\hat{P}^G(z^u, \epsilon_c))$. Desse modo, se $(\tilde{x}^0, \bar{y}^0)$ não for ótima para (P), a solução $(\bar{x}^1, \bar{y}^1)$ obtida terá $\bar{y}^1 \neq \bar{y}^0$, $f(\bar{x}^1, \bar{y}^1) < f(\tilde{x}^0, \bar{y}^0)$ e será utilizada para obter a solução $(\tilde{x}^1, \bar{y}^1)$, que será a melhor solução viável de (P) tendo $\bar{y}^1$ como valor para y. Esta última solução será então utilizada para atualizar o limitante superior z^u. Estendendo esse raciocínio, temos que, a cada iteração, a resolução de $(\hat{P}^G(z^u, \epsilon_c))$ nos fornecerá uma solução inteira para y de valor $\bar{y}^k$ diferente de todos os valores obtidos nas iterações anteriores (isto é, $\bar{y}^k \neq \bar{y}^j, \forall j \in \{0, 1, \ldots, k-1\}$), até que o problema $(\hat{P}^G(z^u, \epsilon_c))$ seja inviável, ou forneça um valor $\bar{y}^k$ não inteiro em sua solução ótima global. Temos ainda que o limitante superior z^u assumirá uma sequência decrescente de valores. Uma vez que o número de soluções diferentes para y é finito, já que todas as variáveis inteiras são binárias, temos que AMGI1 convergirá em um número finito de iterações, e a última solução viável obtida para (P) será ótima para este problema com tolerância ϵ_c. □

Apontamos que o Algoritmo 3.1 não obtém limitantes inferiores para (P) até que

algum dos seus critérios de parada de otimalidade seja atendido, isto é, o problema $(\hat{P}^G(z^u, \epsilon_c))$ se torna inviável ou fornece solução ótima global não inteira. Quando algum destes eventos ocorre, podemos tomar o valor $z^u - \epsilon_c$ como limitante inferior para (P). Em situações práticas onde seja necessário obter uma sequência de limitantes inferiores para (P) antes da otimalidade ser comprovada, pode-se utilizar uma versão modificada do algoritmo. Seja a restrição de corte de nível objetivo

$$f(x, y) \ \leq \ w \tag{3.18}$$

onde w é constante, possivelmente uma estimativa para o limitante inferior. Conforme já mencionado, se esta restrição tornar o problema $(\hat{P}^G(w, 0))$ inviável ou com solução ótima não inteira, podemos tomar w como limitante inferior para (P). De posse desta informação, podemos desenvolver uma adaptação do algoritmo onde o parâmetro w seja atualizado em uma busca binária entre o limitante inferior z^l e o limitante superior z^u corrente, conforme expresso no Algoritmo 3.2. A estimativa inicial de z^l pode ser obtida com a resolução de $(\tilde{P})$ (relaxação contínua de (P)).

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

```
Seja (x̃^0, ỹ^0) a solução de (P̃) ;
z^l = f(x̃^0, ỹ^0) ;
se ỹ^0 for inteira então
    (x*, y*) = (x̃^0, ỹ^0) ;
    z^u = f(x̃^0, ỹ^0) ;
    pare o algoritmo ;
Seja (x̄^1, ȳ^1) uma solução ótima global de (P^G) ;
se ȳ^1 for inteira então
    Seja x̃^1 uma solução ótima de (P(ȳ^1)) ;
    (x*, y*) = (x̃^1, ȳ^1) ;
    z^u = f(x̃^1, ȳ^1) ;
    k = 2 ;
    enquanto z^u − z^l > ε_c faça
        w = (z^l + z^u)/2 ;
        se (P̂^G(w, 0)) for viável então
            Seja (x̄^k, ȳ^k) uma solução ótima global de (P̂^G(w, 0)) ;
            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
                z^l = w ;
        senão
            z^l = w ;
        k = k + 1 ;
```

Algoritmo 3.2: Algoritmo AMGI1 modificado.

Ressaltamos que, no Algoritmo 3.2, o parâmetro w, usado para compor a restrição de corte de nível objetivo (3.15), é calculado como sendo o valor médio entre z^l e z^u (linha 14). Desse modo, podemos garantir que o *gap* de otimalidade $z^u - z^l$ sempre será reduzido a, pelo menos, a metade a cada iteração do laço nas linhas 13-25, visto que, ou limitante inferior z^l será atualizado para w (linhas 22 e 24), ou o limitante superior z^u será atualizado para um valor menor ou igual a w (linha 20). Outros esquemas para a atualização de w podem vir a ser empregados, tentando privilegiar, por exemplo, a atualização de z^u em detrimento da de z^l, ou vice-versa. Uma possível estratégia seria a adoção de um parâmetro $\alpha \in (0,\ 1)$ para o cálculo de w segundo a expressão

$$w = (1 - \alpha)z^l + \alpha z^u \ .$$

O parâmetro α controlaria se o corte de nível objetivo seria construído mais próximo ao limitante inferior ou ao superior.

Embora essa nova versão de AMGI1 possa parecer bastante atrativa à primeira vista, temos observado que, na prática, a resolução dos problemas de minimização de *gap* se torna consideravelmente mais dispendiosa quando o mesmo não fornece solução inteira, o que pode acabar tornando essa versão de AMGI1 menos eficiente em relação à original.

Por fim, apontamos que, embora os problemas de minimização de *gap* possuam função objetivo não convexa, os mesmos possuem particularidades que podem vir a ser empregadas na elaboração de um método especializado para a sua resolução. Tais particularidades incluem:

- Função objetivo quadrática, côncava e separável;
- Região viável convexa;
- Limite inferior conhecido;
- Possibilidade de interromper o processo de resolução destes subproblemas a qualquer momento em que um limitante inferior acima de zero for obtido para os mesmos.

Temos notado ainda que, na prática, pode ser útil considerar a integralidade de y nos problemas de minimização de *gap* com o objetivo de melhorar a eficiência de sua resolução, pois esta informação pode acelerar um possível processo de divisão do espaço feito por algoritmos de otimização global. De um modo geral, estratégias auxiliares que acelerem a resolução dos problemas de minimização de *gap* são de extrema utilidade para AMGI1, visto que o algoritmo tende a gastar algo em torno de 100% de seu tempo de execução resolvendo estes problemas.

Capítulo 4

O Algoritmo de Minimização do *Gap* de Integralidade (versão 2)

4.1 O problema abordado

Neste capítulo, ainda abordamos o problema convexo de Programação Não Linear Inteira Mista (PNLIM) binária:

$$\begin{array}{lll} (P) & \min\limits_{x,\,y} & f(x,y) \\ & \text{sujeito a:} & g(x,y) \;\leq\; 0, \\ & & x \in X,\; y \in Y \cap \{0,\,1\}^{n_y}, \end{array} \tag{4.1}$$

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 e duplamente diferenciáveis e contínuas. Assumimos aqui que (P) possui solução ótima.

4.2 Motivação

Conforme o Teorema 3.1 (Capítulo 3) enuncia, a convergência para a otimalidade de AMGI1 está fortemente fundamentada na obtenção de soluções ótimas globais para os problemas de minimização de *gap* (P^G) e ($\hat{P}^G(z^u, \epsilon_c)$). Embora esses problemas não convexos possuam particularidades que poderiam facilitar o seu tratamento, conforme discutido anteriormente, 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 AMGI1 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) das quais dispomos nos dias de hoje. Denominamos nossa abordagem como *Algoritmo de Minimização do Gap de Integralidade 2*, ou AMGI2.

Segundo nosso melhor conhecimento, a metodologia mais indicada para otimizar os problemas de minimização de *gap* seria a aplicação de um algoritmo de *Branch-And-Bound* Espacial (BBE) . Considerando a aplicação de um algoritmo de BBE básico sobre um dos problemas de minimização de *gap*, no momento apropriado para a divisão do espaço (*branching*), seria necessária a escolha de uma variável y_p para a geração de duas novas subpartes do espaço. Em uma delas seria adotada a restrição $y_p \leq \upsilon$, enquanto na outra seria adotada a restrição $y_p \geq \upsilon$, onde υ seria um número real escolhido no intervalo $(\bar{l}_{y_p},\ \bar{u}_{y_p})$, e $\bar{l}_{y_p}$ e $\bar{u}_{y_p}$ seriam, respectivamente, os limitantes inferior e superior de y_p na parcela do espaço sendo explorada correntemente (inicialmente, $\bar{l}_{y_p} = 0$ e $\bar{u}_{y_p} = 1$).

Observe que as duas parcelas do espaço geradas possuiriam intersecção entre si e sua união formaria a parcela original que lhes deu origem, diferentemente do que ocorre na aplicação do *Branch-And-Bound* em programação inteira. O que gostaríamos de ressaltar é que, embora um algoritmo de BBE básico particione o espaço das variáveis sobre valores υ fracionários, já sabemos de antemão que as soluções de interesse apresentam valores inteiros para as variáveis y, pois essas são as soluções que minimizam nossa função objetivo que mede o *gap* de integralidade. Tendo essa valiosa informação em mente, construímos esta nova versão de AMGI integrando o Algoritmo 3.1 com um procedimento de *Branch-And-Bound* que 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. Nesse contexto, mostra-se oportuna uma breve apresentação do algoritmo tradicional de *Branch-And-Bound*, feita a seguir:

4.3 O Algoritmo de *Branch-And-Bound* padrão

Os algoritmos de *Branch-And-Bound* (BB), juntamente com seus derivados, como *Branch-And-Cut* e *Branch-And-Price*, se constituem numa técnica extremamente fundamental na área de programação inteira, seja no âmbito linear ou não linear. Estes algoritmos se baseiam no particionamento do espaço das variáveis inteiras do problema abordado. No nosso contexto, definimos como $(P_{\bar{Y}})$ o subproblema de (P) referente a uma partição $\bar{Y} \subseteq Y$:

$$
\begin{aligned}
(P_{\bar{Y}}) \qquad \min_{x,\,y} \quad & f(x,y) \\
\text{sujeito a:} \quad & g(x,y) \;\leq\; 0 \\
& x \in X,\; y \in \bar{Y} \cap \{0,\,1\}^{n_y}.
\end{aligned}
\tag{4.2}
$$

Definimos então a relaxação contínua de $(P_{\bar{Y}})$, isto é, o subproblema deste onde as restrições de integralidade são relaxadas, como $(\tilde{P}_{\bar{Y}})$:

$$
\begin{aligned}
(\tilde{P}_{\bar{Y}}) \qquad \min_{x,\,y} \quad & f(x,y) \\
\text{sujeito a:} \quad & g(x,y) \;\leq\; 0 \\
& 0 \leq y \leq 1, \\
& x \in X,\; y \in \bar{Y}.
\end{aligned}
\tag{4.3}
$$

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

```
1  z^u = +∞ ;
2  Y^0 = Y ;
3  Seja N = {0} a lista inicial de nós em aberto;
4  i = 0;
5  Seja L^i o limitante inferior do nó i ;
6  L^0 = −∞ ;
7  enquanto N ≠ ∅ faça
8      z^l = min_j{L^j : j ∈ N} ;
9      Selecione um nó k de N ;
10     se (P̃_{Y^k}) é inviável então
11         N = N\{k} ;                                  // Poda por inviabilidade
12     senão
13         Seja (x̃^k, ỹ^k) uma solução ótima de (P̃_{Y^k}) ;
14         se f(x̃^k, ỹ^k) < z^u − ε_c então
15             se ỹ^k for inteira então
16                 z^u = f(x̃^k, ỹ^k) ;
17                 (x^*, y^*) = (x̃^k, ỹ^k) ;
18                 N = N\{k} ;                          // Poda por otimalidade
19                 N = N\{j : L^j ≥ z^u − ε_c} ;        // Podas por limite
20             senão
                   // Ramificação
21                 Selecione uma variável y_j com valor ỹ^k_j não inteiro ;
22                 Y^{i+1} = Y^k ∩ {y ∈ R^{n_y} : y_j ≤ ⌊ỹ^k_j⌋} ;
23                 Y^{i+2} = Y^k ∩ {y ∈ R^{n_y} : y_j ≥ ⌈ỹ^k_j⌉} ;
24                 L^{i+1} = L^{i+2} = f(x̃^k, ỹ^k) ;
25                 N = N ∪ {i + 1, i + 2}\{k} ;
26                 i = i + 2 ;
27         senão
28             N = N\{k} ;                              // Poda por limite
```

Algoritmo 4.1: Algoritmo de *Branch-And-Bound* básico.

O algoritmo básico de BB, apresentado aqui como Algoritmo 4.1, resolve as relaxações contínuas de $(P_{\bar{Y}})$ referentes às partições do espaço geradas. Note que,

no Algoritmo 4.1, uma árvore de busca é percorrida, onde, a cada nó k explorado, a relaxação contínua ($\tilde{P}_{Y^k}$) do subproblema na partição Y^k é resolvida. Assim, seja z^u o melhor limitante superior corrente e $(\tilde{x}^k, \tilde{y}^k)$ uma solução ótima de ($\tilde{P}_{Y^k}$), caso este seja viável. Temos então quatro possibilidades distintas a considerar:

1. *O problema* ($\tilde{P}_{Y^k}$) *é inviável*: neste caso, podemos descartar todos as subpartições de Y^k (subárvore abaixo do nó atual), pois também serão inviáveis (poda por inviabilidade, linha 11 do Algoritmo 4.1).

2. $f(x^k, y^k) \geq z^u$: neste caso, uma vez que o valor $f(x^k, y^k)$ é um limitante inferior para (P) em toda subpartição de Y^k, temos a certeza de que nenhuma delas apresenta solução que melhore o limitante superior conhecido, e, portanto, podemos descartá-las (poda por limite, linha 28 do Algoritmo 4.1).

3. $f(x^k, y^k) < z^u$ *e* y^k *é inteira*: neste caso, temos uma solução viável melhor que a melhor solução conhecida. Assim, podemos atualizar z^u e também descartar as subpartições de Y^k (poda por otimalidade, linha 18 do Algoritmo 4.1), pois $f(x^k, y^k)$ ainda é um limitante inferior para as mesmas, e, portanto, nenhuma delas pode fornecer solução que melhore o novo limitante superior. Podemos ainda descartar todos as demais partições em aberto da árvore que apresentam limitante inferior maior que o novo valor de z^u (podas por limite, linha 19 do Algoritmo 4.1).

4. $f(x^k, y^k) < z^u$ *e* y^k *não é inteira*: neste caso, precisamos avaliar subpartições de Y^k, pois existe a possibilidade de alguma delas abrigar solução viável que melhore o limitante superior. Escolhemos uma variável y_j para particionar o espaço e ramificamos a árvore, criando assim duas novas subpartições (ramificação, linhas 21-26 do Algoritmo 4.1).

Na prática, costumam-se empregar muitas estratégias auxiliares na aplicação do BB com o intuito de acelerar seu processo de convergência, como, por exemplo, geração de cortes; pré-processamento; (re)inicio promissor (*hot start*); *strong branching*; pseudo-custos; aplicações de heurísticas e ramificação sobre múltiplas variáveis. O leitor interessado em uma discussão mais aprofundada sobre a aplicação de BB para PNLIM, com menção a algumas dessas estratégias, pode consultar os trabalhos [1, 26].

4.4 Algoritmo AMGI2

Definimos então o problema de minimização de *gap* de integralidade com corte de nível objetivo tomado em um determinado nó k:

$$(\hat{P}^G(Y^k, z^u, \epsilon_c)) \qquad \min_{x,\,y} \quad \sum_{i=1}^{n_y} y_i(1-y_i) \tag{4.4}$$

$$\text{sujeito a:} \quad g(x,y) \;\leq\; 0 \tag{4.5}$$

$$f(x,y) \;\leq\; z^u - \epsilon_c \tag{4.6}$$

$$0 \leq y \leq 1 \tag{4.7}$$

$$x \in X,\; y \in Y^k \tag{4.8}$$

onde Y^k é a partição de Y referente ao nó k.

A segunda versão de AMGI foi projetada para ser utilizada 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 ($\bar{x}^k$,$\bar{y}^k$) com $\bar{y}^k$ não inteiro é obtida para um desses problemas, particionamos o espaço ramificando sobre alguma variável y_j com valor $\bar{y}^k_j$ fracionário. Por outro lado, quando obtemos $\bar{y}^k$ inteiro nessas mesmas circunstâncias, utilizamos essa nova solução para obter a solução $(\tilde{x}^k, \bar{y}^k)$ a partir de $(P(\bar{y}^k))$ que atualizará o limitante superior z^u. De posse dessa última, resolvemos novamente o problema de minimização de *gap* na mesma partição corrente k, mas agora com a restrição de corte de nível objetivo atualizada. Repetimos esses passos na partição corrente k até obtermos uma solução ($\bar{x}^k$,$\bar{y}^k$) com $\bar{y}^k$ não inteiro para $(\hat{P}^G(Y^k, z^u, \epsilon_c))$, ou este problema se tornar inviável. Nesse primeiro caso ($\bar{y}^k$ não inteiro), procedemos à ramificação. No segundo caso ($(\hat{P}^G(Y^k, z^u, \epsilon_c))$ inviável), realizamos poda.

O algoritmo de AMGI2 é mostrado como Algoritmo 4.2. 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 em [19] para apresentar um algoritmo para PNLIM baseado no algoritmo de Aproximação Externa de [11, 18], que originalmente pode se valer de várias aplicações sequenciais de BB para a resolução de problemas de PLIM aproximativos. No algoritmo proposto em [19], um procedimento único de BB é utilizado para a resolução de um problema de PLIM aproximativo que então é atualizado dinamicamente ao longo da exploração dos nós.

Ressaltamos que no *Branch-And-Bound* de AMGI2, não dispomos de qualquer artifício para obtenção de limitantes 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

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

1 $z^u = \infty$;
2 $Y^0 = Y$;
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 4.2: Algoritmo AMGI2.

(4.6) provoca inviabilidade em determinadas partições que não possam melhorar o limitante superior, isto é, partições que seriam podadas por limitante em um BB tradicional de programação inteira acabam sendo podadas por inviabilidade em nossa abordagem. De forma similar, quando a melhor solução inteira de uma partição porventura é obtida, esta mesma restrição causará inviabilidade na partição corrente ou em partições descendentes. Assim, podas que ocorreriam por otimalidade em um BB tradicional ocorrem também por inviabilidade aqui. Podas que ocorreriam no BB tradicional por inviabilidade continuam ocorrendo dessa mesma maneira em AMGI2.

Há uma relação evidente entre o algoritmo de *Branch-And-Bound* tradicional

para PNLIM e AMGI2. A principal diferença diz respeito ao fato que um BB 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 BB pode demandar muitas iterações até encontrar alguma solução viável para o problema abordado. Essa demora pode acarretar em um consumo de memória mais alto por parte do procedimento, já que uma lista com nós em aberto precisa ser mantida, e sem nenhuma solução viável conhecida, não é possível eliminar nós dessa lista através de poda por limite. Em alguns casos, a demora na obtenção de soluções viáveis pode provocar a exploração desnecessária de muitos nós que seriam podados caso se conhecesse previamente um bom limitante superior. Por sua vez, AMGI2 otimiza nas partições buscando encontrar alguma solução inteira que melhore o limitante 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 a uma determinada quantidade de tempo computacional ou número máximo de iterações, ou ainda, pará-lo quando a primeira solução viável for encontrada. Esta solução viável poderia inclusive ser utilizada para melhorar o desempenho de um BB tradicional ou de outros algoritmos que resolvam (P).

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*. Esta característica facilita a sua implementação uma vez que, rotinas de PNL locais são mais acessíveis que rotinas de otimização global. 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, para a ramificação, pode-se escolher a variável y_j com valor mais distante da integralidade na solução do problema de minimização de *gap* referente à partição considerada.

4.5 Atribuição de Pesos às Variáveis Inteiras

Os problemas de minimização do *gap* considerados até o presente momento tratam todas as variáveis inteiras como tendo igual peso na função objetivo. Na prática, muitas vezes tem-se o conhecimento prévio de que priorizar a integralidade de um determinado subconjunto de variáveis pode facilitar todo o processo de resolução do

problema, por exemplo, agilizando a obtenção de soluções viáveis ou melhorando o crescimento do limitante inferior das subpartições geradas no processo de ramificação. Em ambos os casos, favorecem-se assim possíveis podas por limite e diminui-se o tempo total de exploração dos nós da árvore de BB. Sob essa ótica, muitos pacotes computacionais de PLIM, por exemplo, permitem que o usuário atribua prioridades para cada variável inteira do problema sendo otimizado. Essas prioridades são levadas em consideração no momento de escolher a variável sob a qual a ramificação é feita.

Tanto AMGI1 quanto AMGI2 podem incorporar possíveis pesos para as variáveis inteiras de uma maneira mais efetiva do que o BB tradicional. Basta para isso, além de tomar as prioridades no momento da ramificação, considerar como função objetivo dos problemas de minimização de *gap* a seguinte expressão:

$$\sum_{i=1}^{n_y} \beta_i y_i (1 - y_i) \tag{4.9}$$

onde $\beta_i \geq 0$ é o peso referente à variável i. Ressaltamos que, considerar $\beta_i = 1, i = 1, \ldots, n_y$ equivale a considerar a função objetivo de (P^G). Note também que é possível adotar $\beta_i = 0$ sem qualquer prejuízo se a variável inteira i porventura já estiver fixa no contexto em questão.

Observamos que considerar a expressão (4.9) juntamente com um esquema dinâmico de atualização dos pesos β_i pode ser particularmente interessante em AMGI2. Por exemplo, se for detectado que, em diversas partições, a solução ótima local obtida apresenta dificuldades em zerar o *gap* de integralidade de uma determinada variável y_i, pode-se passar a adotar um valor de β_i maior que o até então utilizado, para que assim a variável y_i receba maior preferência do modelo para minimizar seu *gap*. Por outro lado, se uma determinada variável y_j alcança facilmente valores inteiros nas soluções ótimas obtidas em cada nó, pode-se diminuir seu peso β_j para tentar incentivar o modelo a zerar o *gap* das demais variáveis inteiras.

Desse modo, no algoritmo AMGI2 (Algoritmo 4.2), a cada nó k explorado, pode-se considerar:

$$\beta_{i,k} \quad = \quad \begin{cases} 0 & \text{, se } y_i \text{ está fixa no nó } k \\ \beta_{i,0} + \gamma\sigma_i & \text{, caso contrário} \end{cases} \tag{4.10}$$

onde γ é um parâmetro não negativo, σ_i é a média do *gap* da variável y_i em todos os nós da árvore já explorados onde y_i não estava fixa, e $\beta_{i,0}$ é o peso inicial para y_i. Pode-se tomar $\beta_{i,0} = 1$, ou utilizar pesos já pré-definidos pelo usuário. Observe que se $\gamma = 0$, recai-se no caso onde cada variável terá o mesmo peso nas funções objetivo dos problemas de minimização de *gap* em todos os nós. Temos observado que essa atualização dinâmica nos pesos β_i pode melhorar o desempenho de AMGI2,

especialmente porque a solução ótima local obtida em cada problema de minimização do *gap* é bastante dependente do algoritmo e da implementação adotada na rotina de PNL local. Desse modo, a atualização dinâmica dos pesos pode ajudar a ganhar diversidade no processo de obtenção dos ótimos locais para os problemas mencionados. Por fim, vale mencionar que essa estratégia de atualização dinâmica dos pesos β_i lembra, de certo modo, a conhecida estratégia comumente adotada em algoritmos de BB denominada como pseudo-custo, embora, estas duas estratégias possuam finalidades diferentes, uma vez que os pseudo-custos são utilizados para a tomada da decisão da variável sobre a qual a ramificação será feita. (O leitor interessado em informações sobre pseudo-custos no âmbito de BB para PNLIM é encorajado a consultar [26].)

4.6 Uso Como Heurística

Uma das características mais marcantes de AMGI2 é a sua agilidade para encontrar soluções viáveis. Deste modo, torna-se natural o seu uso na qualidade de heurística de viabilidade. Uma vez que o processo de enumeração de BB ao qual AMGI2 se baseia pode ser bastante dispendioso, pode ser necessária a adoção de mecanismos que provoquem a parada do algoritmo de forma precoce nesse contexto. Uma estratégia natural a ser empregada seria a interrupção do algoritmo imediatamente após a primeira solução viável ser encontrada. Entretanto, esta ação poderia fazer o algoritmo apresentar uma solução ruim, o que seria desvantajoso. Outras estratégias naturais que poderiam ser empregadas são a adoção de um tempo máximo de execução, ou número máximo de iterações de modo a permitir que o algoritmo encontre uma sequência de soluções viáveis reportando assim a última delas, que, pela definição, é a melhor encontrada. Ainda assim, o ajuste ideal do tempo máximo de execução ou número máximo de iterações pode não ser uma tarefa trivial, visto que esses números podem variar bastante dependendo da instância do problema sendo abordada.

Devido a isso, apresentamos aqui uma estratégia simples que pode ser empregada em AMGI2 de modo a tornar mais prático o seu uso como heurística de viabilidade. A estratégia se baseia no fato de que, num universo utopicamente ideal, a resolução do problema de minimização de *gap* no nó k sempre nos forneceria a solução ótima global. Nessas circunstâncias, se essa solução não satisfizer a integralidade e, simultaneamente, a restrição de corte de nível objetivo estiver ativa sobre a mesma, poderíamos podar esse nó por limite pois não haveria solução que zerasse o *gap* abaixo do limitante superior corrente. Isso, por suposto, se realmente tivéssemos a certeza de que a solução em mãos é de fato a ótima global para o problema de minimização do *gap* sendo considerado. Sendo assim, uma vez utilizando AMGI2

na qualidade de heurística, isto é, sem o compromisso estrito com a otimalidade, podemos descartar os nós aos quais obtivermos solução para o problema de minimização de *gap* onde o corte de nível objetivo esteja ativo. Observe que, para que isto ocorra, é necessário que AMGI2 já tenha encontrado ao menos uma solução viável. Desse modo, mesmo que partições sejam podadas indevidamente, AMGI2 apresentará uma solução viável ao final da execução. A tendência é que este novo critério de poda de fato elimine muitas partições precocemente (algumas até indevidamente) e, assim, a execução do algoritmo chegaria ao seu final de modo mais rápido. Dessa forma, AMGI2 poderá ter sua execução acelerada, na expectativa de atender ao requisito implícito de que heurísticas de viabilidade devem fornecer soluções viáveis com baixo tempo computacional. Esta última estratégia poderia ainda ser empregada juntamente com a imposição de um tempo máximo de execução ou número máximo de iterações todavia.

Capítulo 5

Resultados e Contribuições Computacionais

5.1 Resultados Computacionais

Para a realização de testes computacionais, consideramos um conjunto padrão de 152 instâncias de problemas (convexos) de PNLIM binária disponível em [43]. Informações adicionais sobre essas instâncias podem ser encontradas no Apêndice A. Os testes rodaram em um computador com processador core i7 4790 (3,6 GHz), 16 GB de memória RAM sob o sistema operacional Open Suse Linux 13.1 [44]. Com exceção de AMGI1, realizamos implementações próprias de todos os demais algoritmos mencionados nesse Capítulo (exatos e heurísticos) através da linguagem C++ utilizando o compilador `ICPC 16.0.0` [45]. Devido à dificuldade em se obter boas rotinas de otimização global para implementações em C++, o algoritmo AMGI1 foi totalmente implementado no ambiente `GAMS 24.5.3` [46]. Para a resolução de problemas de PLIM exigidos por algoritmos que se valem de aproximação linear, foi utilizado o pacote `Cplex 12.6.0.0` [47]. Para a resolução de subproblemas de PNL convexos, foi adotado o pacote `Mosek 7.1.0.33` [48] e para a resolução dos problemas de minimização de *gap* de AMGI2, foi adotado o pacote `Ipopt 3.12.4` [49] com `HSL 2014.01.10` [50], uma vez que não é possível utilizar `Mosek` em problemas não convexos, ainda que seja apenas para obtenção de ótimos locais. 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.

5.1.1 Algoritmos Exatos

Neste primeiro contexto de execução, consideramos os seguintes algoritmos exatos para PNLIM:

- AMGI1 (versão original apresentada no Algoritmo 3.1);
- AMGI2 OBJ FIX (Algoritmo 4.2 sem pesos na função objetivo);
- AMGI2 OBJ DIN (Algoritmo 4.2 com pesos na função objetivo conforme Expressão (4.10));
- *Branch-And-Bound* puro (BB) com pesudo-custos conforme [26];
- Plano de Corte Estendido (PCE) de Westerlund e Pettersson [20];
- Aproximação Externa (AE), proposto por Duran e Grossmann [11] e aperfeiçoado por Fletcher e Leyffer [18];
- *Branch-And-Bound* híbrido com Aproximação Externa (BB+AE HIB) de Melo, Fampa e Raupp [1, 36].

O tempo máximo de execução de cada algoritmo sobre cada instância de teste foi limitado a 4 horas. Com exceção de AMGI1, os demais algoritmos aqui mencionados contaram com a aplicação de uma rotina de pré-processamento própria cujo objetivo é diminuir a largura das caixas das variáveis, isto é, aumentar os limitantes inferiores e/ou diminuir os limitantes superiores das mesmas. Esta rotina de pré-processamento é aplicada no início da execução de cada algoritmo sobre possíveis restrições lineares presentes no problema sendo tratado. Nos algoritmos que realizam particionamento do espaço (AMGI2, BB e BB+AE HIB), esta rotina também é utilizada em cada partição do espaço sendo explorada por estes. No algoritmo AMGI1, as rotinas de pré-processamento padrão do ambiente `GAMS` foram adotadas. As abordagens em geral utilizaram tolerância de convergência absoluta e relativa da ordem de 10^{-4}. O parâmetro γ utilizado por AMGI OBJ DIN na formulação da função objetivo dos problemas de minimização de *gap* (Equação (4.10)) foi fixado em 10. Pontos aleatórios são utilizados como solução inicial dos problemas de minimização de *gap* em AMGI2. Ressaltamos que toda execução de AMGI2 sempre utiliza a mesma semente de geração de números pseudo-aleatórios para gerar essas soluções iniciais. Desse modo, diferentes execuções desse algoritmo com os mesmos parâmetros de entrada sobre uma mesma instância sempre fornecem o mesmo resultado.

A Tabela 5.1 traz o número de instâncias onde ao menos uma solução viável foi encontrada por cada algoritmo (coluna # sol viável), o número de instâncias onde cada algoritmo apresentou a melhor solução dentre todas as abordagens (coluna # melhor sol) e o número de instâncias resolvidas por cada algoritmo até a otimalidade (coluna # sol ótima). Pode-se notar que, embora nossas versões de AMGI tenham atestado otimalidade para um número menor de instâncias que os demais

Tabela 5.1: Número de soluções viáveis, de melhores soluções dentre as abordagens e de soluções ótimas (comprovadas) alcançados por algoritmos exatos para PNLIM.

algoritmo	**# sol viável**	**# melhor sol**	**# sol ótima**
AMGI1	149	130	113
AMGI2 OBJ FIX	151	127	105
AMGI2 OBJ DIN	151	134	104
BB	148	120	123
PCE	145	141	142
AE	148	144	144
BB+AE HIB	149	145	144

algoritmos, as mesmas obtiveram os melhores valores com relação ao número de instâncias onde ao menos uma solução viável foi encontrada. AMGI1 foi capaz de encontrar solução viável para 149 das 152 instâncias (mesmo número alcançado por BB+AE HIB). As duas versões de AMGI2 encontraram soluções viáveis para 151 dos 152 problemas. Estes números evidenciam uma suposta facilidade por parte dos algoritmos AMGI em obter soluções viáveis. Destacamos ainda que, dentre todos os algoritmos considerados aqui, apenas as duas variantes de AMGI2 foram capazes de fornecer soluções viáveis para as instâncias `trimloss6` e `trimloss7`, que possuem, cada uma, 56 variáveis contínuas, 289 variáveis binárias, 147 restrições lineares e 7 restrições não lineares, conforme pode ser verificado no Apêndice A.

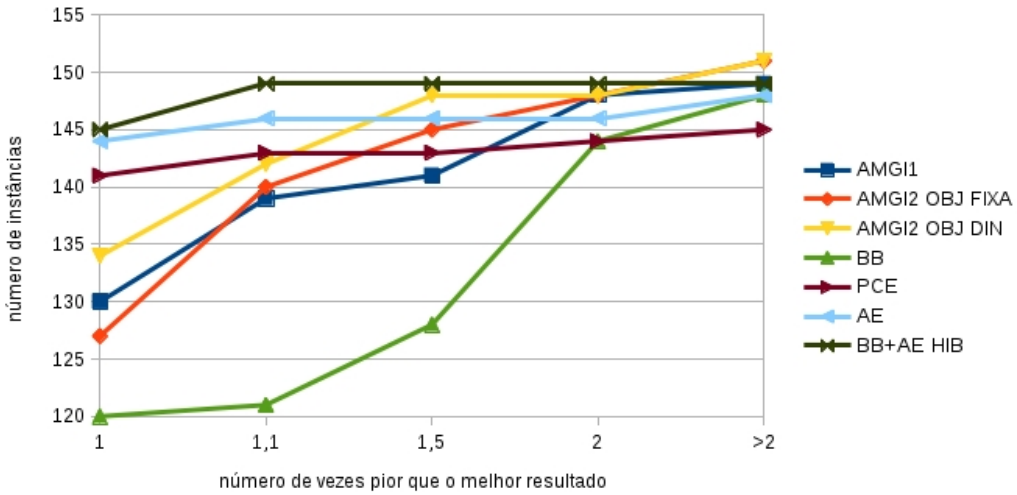


Figura 5.1: Comparação de algoritmos exatos para PNLIM - melhor solução encontrada.

A Figura 5.1 apresenta um gráfico comparativo entre as abordagens com relação à qualidade da melhor solução encontrada. Note que os dados estão normalizados em relação ao melhor resultado obtido por todas as abordagens sobre cada instância. No eixo das abscissas (x), temos valores discriminados para o número de vezes

pior que o melhor resultado apresentado dentre todos os algoritmos. No eixo das ordenadas, temos o número de instâncias alcançado por cada abordagem, isto é, o número de instâncias para as quais o resultado obtido por cada algoritmo é menor ou igual a x vezes o melhor resultado. Note, por exemplo, que a curva de AMGI1 passa pelo ponto (1, 130), significando que, AMGI1 foi capaz for fornecer a melhor solução (1 vez pior que o melhor resultado) para 130 instâncias. Em seguida, a curva passa pelo ponto (1,1, 139), indicando que, para 139 instâncias, AMGI1 foi capaz de fornecer uma solução até 10% acima da melhor solução encontrada entre as abordagens (1,1 vez pior que o melhor resultado). Estendendo esse raciocínio, percebemos que AMGI1 encontra solução até 50% acima do melhor resultado para 141 instâncias e até 100% acima do melhor resultado para 148 instâncias. Por fim, o gráfico informa que quando consideramos soluções acima de 100% do melhor resultado, AMGI1 fornece solução para 149 instâncias. Observe, que, 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.

O gráfico da Figura 5.1 indica que a abordagem BB+AE HIB obteve os melhores resultados quanto à qualidade da solução final apresentada. Apontamos que este algoritmo consegue se valer tanto das vantagens de um algoritmo de *Branch-And-Bound* quanto das vantagens do algoritmo de Aproximação Externa (pertencente à classe de aproximação linear), uma vez que se constitui em um híbrido de ambos. Desse modo, este algoritmo ainda pode se aproveitar de todo o ferramental de um pacote avançado de PLIM como `Cplex`, sem ficar excessivamente refém de aproximações lineares que podem não funcionar muito bem em alguns casos de problemas de PNLIM. O gráfico indica ainda que as três abordagens AMGI foram também superadas por AE e PCE (algoritmos de aproximação linear) quando se consideram soluções até 10% acima do melhor resultado. Este cenário começa então a se inverter a partir do momento em que se consideram soluções até 50% acima do melhor resultado. A figura ainda deixa claro que as abordagens AMGI superaram completamente o desempenho do algoritmo de *Branch-And-Bound* puro quanto a qualidade da melhor solução encontrada. Desse modo, temos aqui um bom indício de que as modificações trazidas por AMGI2 podem melhorar consideravelmente a eficácia de um algoritmo de particionamento de espaço como *Branch-And-Bound*.

A Figura 5.1 sugere que, embora tenha comprovado otimalidade em um número relativamente baixo de instâncias, no geral, as soluções encontradas pelos algoritmos AMGI apresentaram boa qualidade como se pode notar observando o número de instâncias onde a solução apontada ficou 1,1 vez pior que o melhor resultado (10% acima da melhor solução). Este número foi de 139 instâncias para AMGI1, 140 para AMGI2 OBJ FIX e 142 para AMGI2 OBJ DIN. Podemos então notar que a

adição de pesos dinâmicos na função objetivo dos problemas de minimização de *gap* trouxeram uma ligeira melhora para AMGI2 OBJ DIN.

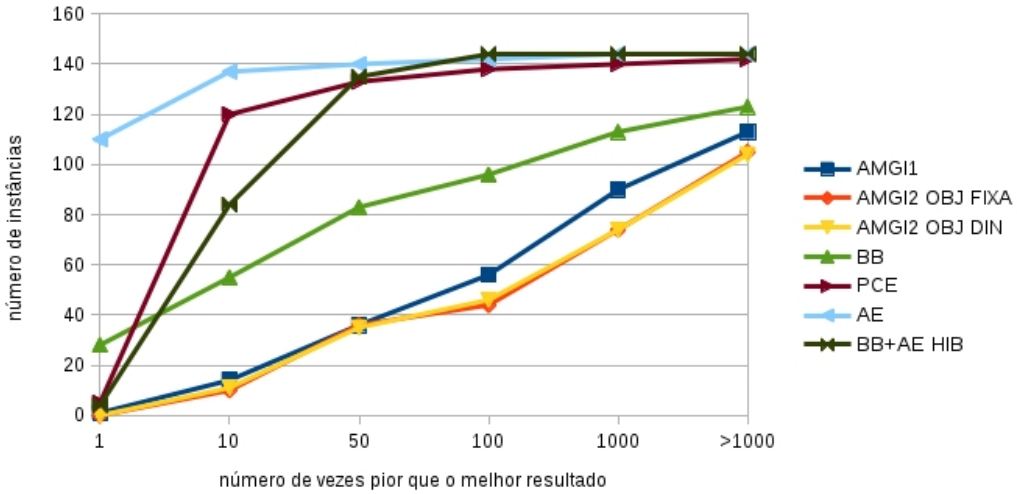


Figura 5.2: Comparação de algoritmos exatos para PNLIM - tempo total de execução para as instâncias resolvidas até a otimalidade.

A Figura 5.2 apresenta um gráfico comparativo entre as abordagens com relação ao tempo total de execução despendido nas instâncias resolvidas até a otimalidade. Note que este gráfico foi construído de forma similar ao gráfico da Figura 5.1. Percebemos que os melhores resultados foram obtidos pelo algoritmo de Aproximação Externa, que é da classe dos algoritmos de aproximações lineares. Isto se deve principalmente ao fato de que este algoritmo se beneficia diretamente da alta eficiência provida pelo pacote `Cplex` na resolução das relaxações lineares dos problemas abordados. Destacamos aqui que o conjunto de instâncias tomado possui grau de não linearidade relativamente baixo, com a maioria das instâncias possuindo um número de restrições lineares consideravelmente maior que o de não lineares, o que favorece os algoritmos de aproximação linear. Para apoiar essa tese, podemos mencionar que o algoritmo PCE, que também é de aproximação linear, foi o que teve a segunda melhor eficiência dentre os considerados. Por não ser totalmente baseado em relaxações lineares dos problemas, o algoritmo BB+AE HIB não conseguiu repetir o bom desempenho do quesito anterior aqui. Este último algoritmo está mais fundamentado nas relaxações contínuas das partições do espaço geradas, eventualmente selecionando algumas dessas partições para aplicação de relaxação linear. O grande número de problemas de PNL que precisam ser resolvidos acaba em alguns casos por fazer esta abordagem possuir eficiência inferior ao algoritmo de Aproximação Externa puro. Para dar suporte a essa tese, temos o fato de que o algoritmo de *Branch-And-Bound* puro apresentou desempenho abaixo de BB+AE HIB no quesito em questão.

Com relação ao tempo de execução, podemos destacar que as três versões de AMGI foram totalmente superadas pelas demais abordagens, tendo AMGI1 se saído levemente superior às versões de AMGI2. Este último fato pode ser parcialmente explicado por uma dificuldade marcante do pacote `Ipopt` em resolver os problemas de minimização de *gap* de AMGI2. Frequentemente `Ipopt` atinge o número máximo de iterações ao abordar estes problemas, mesmo quando dobramos ou triplicamos o número máximo de iterações inicialmente estipulado como padrão pelo *solver*. Ademais, `Ipopt` ainda não possui ferramentas que poderiam melhorar sua eficiência, como discriminação de restrições lineares e quadráticas das demais e armazenamento interno de coeficientes de matrizes esparsas *a priori* de seu processo de otimização. Diante dessas circunstâncias, temos a mais absoluta convicção de que o desenvolvimento de métodos especializados para abordar os problemas de minimização de *gap* melhoraria consideravelmente a performance dos algoritmos AMGI, e, por consequência, aumentaria a sua competitividade em relação aos demais, visto que ficou evidenciado que nossa metodologia é efetiva em encontrar boas soluções. Estes algoritmos especializados poderiam se valer, por exemplo, das características especiais presentes nos problemas de minimização de *gap* mencionadas na Seção 3.3. De toda forma, é valido ainda como perspectiva futura de trabalho ao menos o teste de AMGI2 com outros *solvers* de PNL que utilizem algoritmos diferentes para a resolução dos problemas de minimização de *gap*, dado o insucesso apresentado por parte do algoritmo de pontos interiores implementado em `Ipopt` sobre esses problemas.

Vale destacar ainda que novos estudos e testes ainda precisam ser feitos com o objetivo de melhorar o ajuste e o desempenho das versões de AMGI, cujas implementações aqui ainda se encontram em fase inicial. As implementações dos demais algoritmos, por sua vez, já estão em estágio mais avançado, o que também ajuda a explicar a diferença de performance notada aqui. Podemos mencionar, por exemplo, que diversas estratégias que comumente melhoram o desempenho de algoritmos de BB foram propostas posteriormente a este. Fica então apontado, como perspectiva de trabalho futuro, a busca por mecanismos que também possam melhorar o desempenho dos algoritmos AMGI de modo a torná-los mais competitivos em relação aos demais. Uma possível fonte de inspiração são as estratégias já apresentadas para a melhora de algoritmos de BB.

A Figura 5.3 compara as abordagens quanto à qualidade da primeira solução viável obtida. O gráfico mostra que AE e BB+AE HIB tiveram desempenho bastante similar, o que pode ser um indicativo de que a primeira solução obtida por BB+AE HIB venha da aplicação de AE sobre o problema original abordado, já que BB+AE HIB considera a aplicação periódica e recorrente de AE sobre o problema tratado por tempo limitado. Também é perceptível que as abordagens AMGI possuem desempenho similar ao do algoritmo de *Branch-And-Bound* puro e que o al-

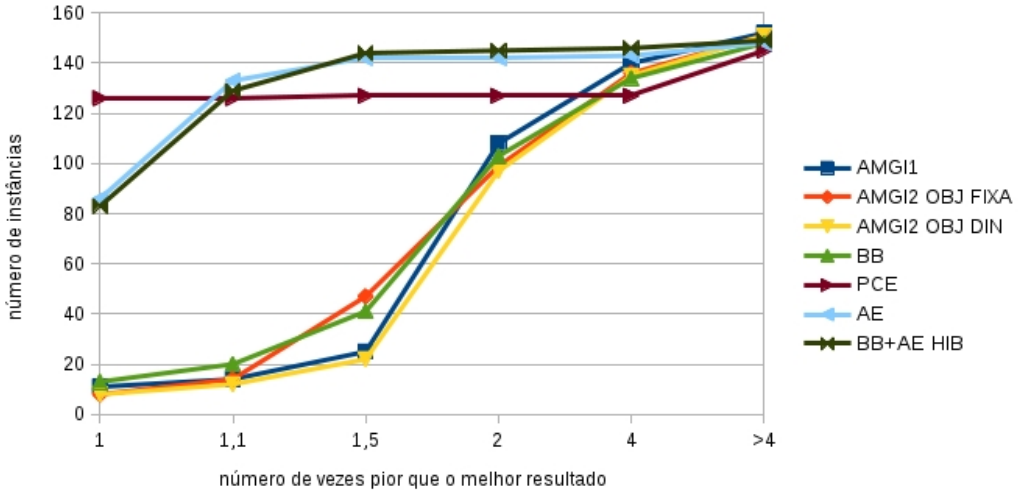


Figura 5.3: Comparação de algoritmos exatos para PNLIM - primeira solução viável encontrada.

goritmo PCE, que é totalmente baseado em planos de corte sobre relaxações lineares, apresentou um bom desempenho nesse contexto. Temos observado que este último algoritmo, em geral, obtém boas primeiras soluções viáveis, ao custo de demandar mais tempo e iterações que o algoritmo AE, que também é baseado em relaxações lineares, demanda para desempenhar esta mesma tarefa. Esta característica pode ser parcialmente observada na Figura 5.4 a seguir.

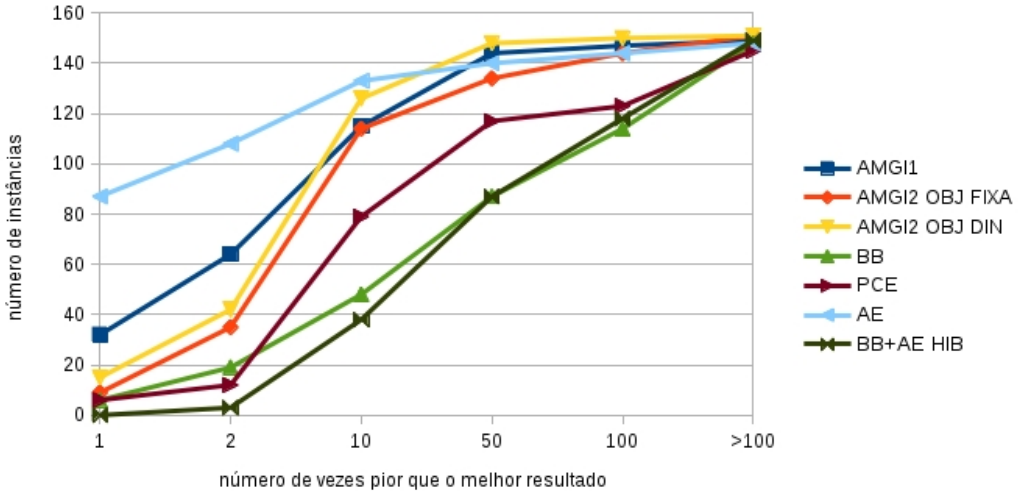


Figura 5.4: Comparação de algoritmos exatos para PNLIM - tempo de execução até a obtenção da primeira solução viável.

A Figura 5.4 compara as abordagens quanto ao tempo de execução necessário para obtenção da primeira solução viável. AE apresentou o melhor desempenho nesse quesito, como já era de se esperar pela análise da Figura 5.2. Entretanto, aqui vale destacar que as abordagens AMGI apresentaram desempenhos semelhantes

entre sí tendo dominado completamente os algoritmos BB, PCE e BB+AE HIB. Uma avaliação conjunta das Figuras 5.3 e 5.4 nos aponta que AMGI2 foi capaz de diminuir sensivelmente o tempo que o algoritmo de BB demanda para encontrar a primeira solução viável, sem sofrer perdas no que diz respeito à qualidade dessa solução. Esse apontamento indica um possível caminho de pesquisa futura, onde aplicação de AMGI2 e de BB podem ser integradas. Pode-se, por exemplo, começar a resolução do problema por AMGI2 até a primeira solução viável ser encontrada, ou até a execução de p iterações de AMGI2 sem melhora da melhor solução conhecida, e, então, passar para a aplicação de BB aproveitando desde já a lista de nós em aberto deixada por AMGI2 no momento de sua interrupção.

5.1.2 Algoritmos Heurísticos

Nesta subseção, comparamos o desempenho das seguintes heurísticas no contexto de PNLIM binária:

- AMGI2 versão heurística, com peso dinâmico na função objetivo (Equação (4.10)) e poda por limite (descrita na Seção 4.6);
- *Feasibility Pump* (FP) para PNLIM apresentado por Bonami e Gonçalves [37];
- *Feasibility Pump* baseado em Aproximação Externa (FP-AE) de Bonami *et al.* [38];
- Heurística de Mergulho de Bonami e Gonçalves [37].

Neste contexto, o tempo máximo de execução dos algoritmos sobre cada instância foi ajustado em apenas 10 minutos, visto que espera-se que heurísticas consigam apresentar solução viável utilizando-se de baixo tempo de execução. Os demais parâmetros de AMGI2 foram ajustados conforme a Subseção 5.1.1.

Tabela 5.2: Número de soluções viáveis, de melhores soluções dentre as abordagens e de soluções ótimas (não comprovadas) alcançados por algoritmos heurísticos para PNLIM.

algoritmo	**# sol viável**	**# melhor sol**	**# sol ótima**
AMGI2	151	105	45
FP	145	30	16
FP-AE	150	35	16
Mergulho	128	51	17

A Tabela 5.2 traz o número de instâncias onde cada heurística encontrou solução viável (coluna # sol viável), o número de instâncias onde cada heurística apresentou

a melhor solução dentre todas as abordagens (coluna # melhor sol) e o número de instâncias onde a solução ótima por ventura foi obtida (coluna # sol ótima). Pode-se facilmente observar que AMGI2 apresentou os melhores resultados para as três colunas numéricas da tabela. As quantidades de melhores soluções dentre as abordagens e de soluções ótimas (não comprovadas) obtidas por AMGI2 são notavelmente superiores às dos demais algoritmos.

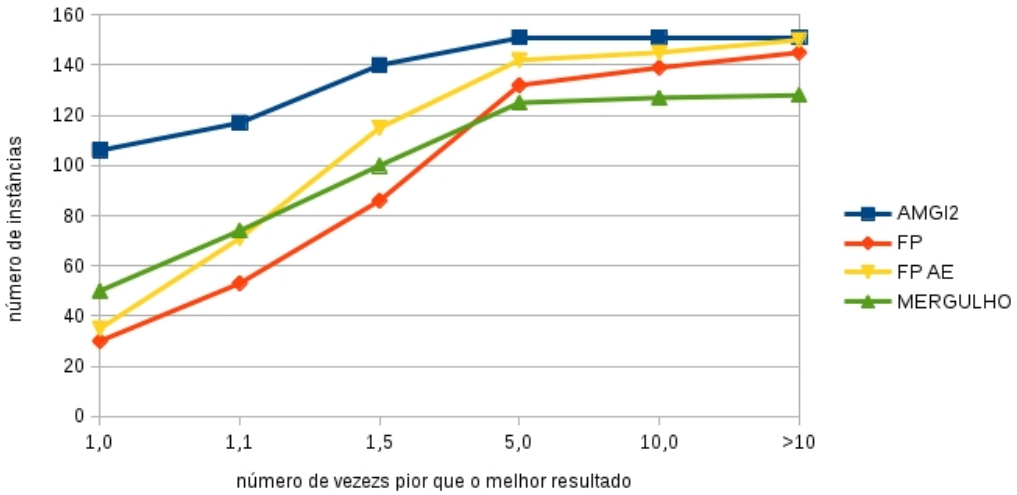


Figura 5.5: Comparação de algoritmos heurísticos para PNLIM - melhor solução encontrada.

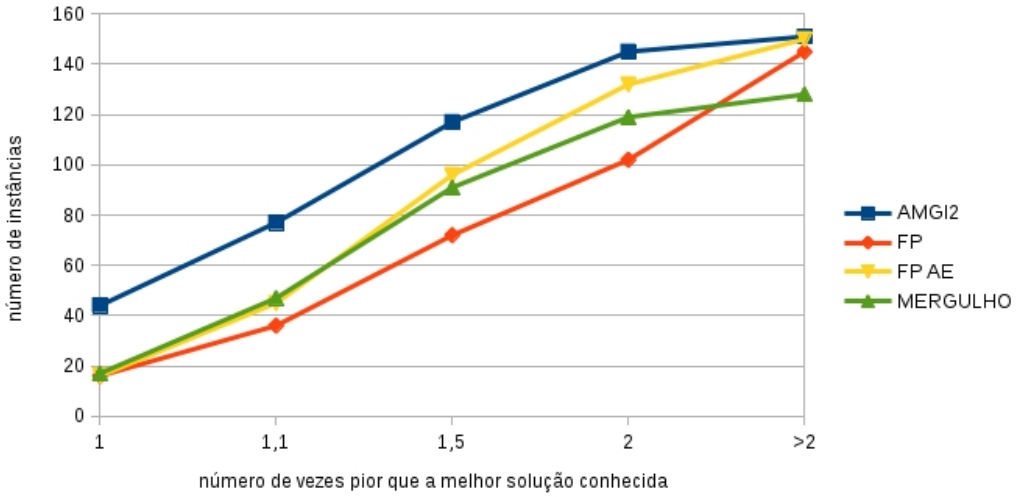


Figura 5.6: Comparação de algoritmos heurísticos para PNLIM - melhor solução encontrada com a melhor solução conhecida.

A Figura 5.5 apresenta o gráfico comparativo entre as abordagens quanto à solução apresentada. Indiscutivelmente, AMGI2 supera as demais heurísticas quanto à qualidade da solução apresentada e ao número de instâncias onde alguma solução viável foi encontrada (151 instâncias). A Figura 5.6 compara as soluções obtidas

pelas abordagens com a solução ótima (ou a melhor solução conhecida) de cada instância. Novamente, o desempenho de AMGI2 é melhor que os das demais heurísticas. Estes resultados indicam que o uso de AMGI2 pode ser muito vantajoso em situações onde partir de boas soluções viáveis seja crucial ou oportuno.

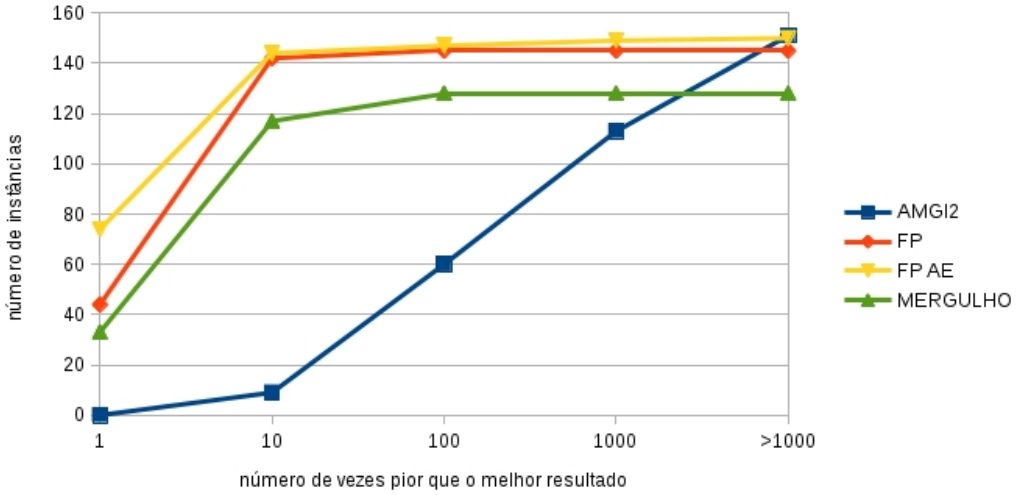


Figura 5.7: Comparação de algoritmos heurísticos para PNLIM - tempo de execução para instâncias onde alguma solução viável foi obtida.

A Figura 5.7 apresenta o gráfico comparativo das abordagens em relação ao tempo de execução. O gráfico mostra que o algoritmo FP-AE obteve os melhores resultados em termos de eficiência. A incorporação de aproximações lineares para prover soluções às variáveis inteiras feita por FP-AE parece ter trazido vantagens ao algoritmo em relação ao arredondamento simples utilizado por FP. Pode-se observar que o desempenho de AMGI2 está notavelmente abaixo das demais heurísticas. Isto se deve em muito à demora por parte de `Ipopt` em resolver os problemas de minimização de *gap* e ao fato de que as demais heurísticas param na primeira solução viável encontrada por não terem meios de melhorar essa solução. AMGI2 por sua vez, possui um mecanismo que lhe permite prosseguir após a obtenção de uma solução, na expectativa de alcançar soluções viáveis ainda melhores. Desse modo, AMGI2 demanda tempo computacional maior que as demais heurísticas, mas, conforme indicado nas Figuras 5.5 e 5.6, pode conseguir resultados expressivamente melhores. Vale ressaltar ainda que, embora os tempos de execução de AMGI2 sejam relativamente grandes, ainda assim, seus resultados foram obtidos respeitando o tempo máximo de apenas 10 minutos em cada instância de teste.

5.2 Contribuições Computacionais

Além de propor novas abordagens para PNLIM (contribuição teórica), este trabalho sempre teve como um de seus objetivos principais o desenvolvimento de um novo pacote computacional aberto para PNLIM (contribuição computacional). Este novo *solver*, denominado como `Muriqui`, tem como missão agregar todo o conhecimento e experiência que obtivemos ao longo desses anos de pesquisa bem como o resultado de trabalhos futuros que ainda serão conduzidos. O desenvolvimento de `Muriqui` tem se mostrado um desafio a cada dia mais complexo, que transcende o escopo desta tese de doutorado. Suas motivações principais são:

- A potencialização da utilidade deste trabalho de pesquisa;
- A ampliação do nosso conhecimento prático sobre a implementação e o funcionamento dos algoritmos estudados;
- O provimento de suporte para o trabalho de outros pesquisadores na área, contribuindo assim para o avanço da ciência como um todo;
- O incentivo ao aumento das aplicações práticas de PNLIM;
- A inexistência de um pacote computacional consolidado para PNLIM, embora já existam alguns disponíveis, em sua maioria de caráter experimental;
- A existência de "lacunas", isto é, funcionalidades úteis ainda não disponíveis, nos pacotes para PNLIM já existentes;
- O incentivo ao desenvolvimento da computação científica brasileira.

`Muriqui` tem por objetivo ser um pacote altamente customizável que faz uso das melhores rotinas computacionais livres e comerciais às quais temos acesso. Além de apresentar uma biblioteca de rotinas para programação em C++, `Muriqui` pode ser utilizado por meio do sistema `AMPL` [51], o que lhe permite alcançar usuários com pouca ou nenhuma habilidade em programação. Os diversos algoritmos mencionados nesse capítulo já estão disponíveis para uso no *software*, a saber:

- Algoritmos exatos:

  Aproximação Externa [11, 18];

  Plano de Corte Estendido original [20];

  Plano de Corte Estendido melhorado (ainda a ser apresentado);

  Branch-And-Bound paralelo;

  Branch-And-Bound paralelo híbrido com Aproximação Externa [1, 36];

Algoritmo de Minimização de *Gap* de Integralidade 2 (apresentado aqui), com suporte à execução paralela por múltiplos processadores;

- Algoritmos heurísticos

  Feasibility Pump [37];

  Feasibility Pump baseado em Aproximação Externa [38];

  Heurística de Mergulho [37];

  Algoritmo de Minimização de *Gap* de Integralidade 2 (apresentado aqui), com suporte à execução paralela por múltiplos processadores.

A disponibilização de diferentes algoritmos no pacote o torna bastante versátil para a abordagem de toda uma variedade de aplicações práticas que podem ter características bastantes distintas. Futuramente, outros algoritmos poderão ser incorporados ao pacote, incluindo métodos exatos, heurísticas e até metaheurísticas, como, por exemplo [52–54]. O desenvolvimento de `Muriqui` já resultou na proposição do algoritmo de *Branch-And-Bound* híbrido com Aproximação Externa, apresentado em [1, 36]. Conforme pode ser conferido na Subseção 5.1.1, esta abordagem tem sido efetiva em diversos problemas de PNLIM. Ademais, a implementação de *Branch-And-Bound* de `Muriqui` permite a customização deste algoritmo para aproveitamento de particularidades presentes em determinadas aplicações. Um exemplo de sucesso na customização do BB de `Muriqui` está descrito em [55], onde características especiais presentes no problema de Steiner Euclidiano foram tomadas para compor um algoritmo de BB otimizado para a resolução deste problema, com base na implementação disponível em `Muriqui`. Os autores reportam altos níveis de ganho de perfomance com a customização desenvolvida para este problema a partir do BB de `Muriqui`, o que abre mais uma linha de atuação para esse pacote computacional.

As implementações dos componentes de `Muriqui`, feitas em C++, atualmente contam com cerca de 40 mil linhas de código em seus arquivos fontes (excluindo linhas vazias e de comentários). Alguns desses componentes já se tornaram projetos independentes que podem agora ser facilmente reutilizados em outras aplicações computacionais, o que se constitui em mais uma contribuição deste trabalho. Estes componentes incluem:

- Um *framework* para implementação genérica de algoritmos de *Branch-And-Bound* com suporte a execução de múltiplas *threads* de processamento para resolução de subproblemas em paralelo;

- Uma Interface de Programação de Aplicações (*Application Programming Interface*) para a utilização de diferentes pacotes de otimização computacional de forma unificada;

- Uma biblioteca para representação genérica de problemas de Programação Não Linear Inteira Mista, com suporte a recebimento de dados via sistema `AMPL`
- Uma biblioteca para armazenamento de coeficientes de matrizes esparsas.

Vale destacar que os componentes acima já tem sido reutilizados com sucesso no desenvolvimento da aplicação computacional referente a outra parte deste trabalho, o pacote computacional `iquad`, cuja descrição é feita na Seção 8.2. O trabalho de implementação aqui mencionado foi totalmente desenvolvido com o intuito de torná-lo útil para pesquisas e aplicações futuras, não apenas de nossa parte, mas também por parte de outros pesquisadores e técnicos nos mais variados campos do conhecimento.

Parte II

Uma Abordagem de *Branch-And-Bound* Espacial para Programação Quadrática Não Convexa

Capítulo 6

Introdução

Problemas de Programação Não Linear (PNL) não convexa, comumente denominados como problemas de otimização global, são problemas de otimização que incorporam funções não convexas na função objetivo sendo minimizada [1] e/ou nas restrições (na forma de desigualdades com limitante superior). Nessa parte do trabalho, estamos interessados na resolução do seguinte problema de otimização global onde a não convexidade se remete apenas às funções quadráticas:

$$\begin{array}{lll} (PQR) \quad z := \min\limits_{x} & f_0(x) \ + \ q_0(x), & \\ \text{sujeito a:} & f_i(x) \ + \ q_i(x) \leq b_i, \quad i = 1, \ldots, m & \\ & l_x \leq x \leq u_x, & \\ & x_j \in \mathbb{Z}, \text{ para } j \in \mathcal{I}, & \end{array} \tag{6.1}$$

onde, para $j = 0, \ldots, m$, $f_j(x)$ são funções não lineares convexas e $q_j(x) = \frac{1}{2} x' Q_j x$ são funções quadráticas. Assumimos aqui que a região viável de (PQR) é limitada e que ao menos alguma das matrizes simétricas $Q_j, \ j = 0, \ldots, m$ não é semidefinida positiva, o que significa dizer que, ao menos uma das funções $q_j(x)$ é não convexa. Note que algumas das matrizes Q_j ou mesmo as funções f_j podem ser nulas. Observe ainda que termos lineares podem estar presentes nestas últimas funções de forma exclusiva ou em conjunto com termos não lineares convexos, e que possíveis restrições de igualdade podem ser desmembradas em duas desigualdades para atender ao formato acima. Aqui, exceto quando explicitamente dito o contrário, vetores são representados como vetores coluna.

A presença de funções não convexas pode acarretar na existência de múltiplas soluções ótimas locais na região viável de um problema de otimização, o que dificulta de forma acentuada o processo de obtenção de alguma solução ótima global. Métodos tradicionais de PNL se baseiam em otimalidade local, o que implica que eles podem ser facilmente levados a convergir para pontos de ótimos locais que não otimizem

[1] para o caso de maximização, a função considerada é não côncava

o problema de forma global. Para tornar a situação ainda mais crítica, em geral, esses métodos não são capazes de atestar se uma solução ótima obtida é global ou apenas local. Diante desse fato, a área de otimização global tem sido tema recorrente de pesquisa (o leitor interessado em revisões bibliográficas gerais pode consultar [56–58]). Os algoritmos desenvolvidos nesse âmbito, em geral, requerem esforço computacional bem mais elevado em comparação aos métodos tradicionais de PNL local, e, em geral, são capazes de resolver em tempo hábil apenas problemas de dimensão reduzida em comparação com estes últimos. Por essa razão, pesquisas pela melhoria e desenvolvimento de novas abordagens para otimização global são de fundamental importância, especialmente porque, no mundo real, muitos problemas práticos podem não ser modelados de forma satisfatória apenas com o uso de funções convexas.

Modelos de otimização envolvendo funções quadráticas não convexas, em particular, tem larga aplicação em otimização combinatória. O problema de corte máximo (*max-cut*) em grafos com pesos pode ser facilmente modelado como um problema quadrático não convexo (com matriz Q_0 indefinida), e seus métodos de solução mais conhecidos exploram esse modelo (o leitor interessado por consultar [59], por exemplo). Adicionalmente, o problema de alocação quadrática (*quadratic assignment problem*), que tem sido utilizado para modelar muitos outros problemas de otimização combinatória estruturados, é diretamente modelado utilizando um modelo quadrático não convexo. (Veja [60], por exemplo). No universo contínuo, minimização de funções côncavas aparecem naturalmente quando temos economias de escala. No campo de Programação Não Linear Inteira Mista (MINLP), podemos mencionar modelos quadráticos não convexos envolvendo termos bilineares, que aparecem em problemas de corte de estoque (para mais informações, consulte [61]), ou mesmo os problemas de minimização de *gap* adotados pelas abordagens AMGI (descritas na Parte I deste texto), que minimizam uma função objetivo quadrática côncava sujeita a restrições convexas.

A metodologia de *Branch-And-Bound* Espacial (BBE) é bastante difundida para a resolução de problemas de otimização global. Esta metodologia consiste em dividir o espaço em busca de soluções viáveis que possam melhorar o limitante superior conhecido (no caso de função objetivo de minimização) para o problema tratado, até que seja comprovado que uma solução viável obtida é a solução ótima global do problema em questão (assumindo, é claro, que esta solução ótima global exista). Para que esse "certificado" de otimalidade possa ser obtido, torna-se crucial a construção de uma relaxação convexa para os subproblemas em cada subporção do espaço explorada. Através da resolução destas relaxações convexas, obtém-se limitantes inferiores para as diferentes subporções do espaço, o que permite o descarte das mesmas tão logo fique comprovado que não possam abrigar soluções melhores do

que a melhor solução conhecida. Isto ocorre quando as respectivas relaxações convexas fornecem limitantes inferiores maiores ou iguais ao limitante superior global do problema (obtido, em geral, com a melhor solução viável encontrada), ou quando estas mesmas relaxações se mostram inviáveis. Em geral, quanto mais forte é a relaxação obtida, isto é, quanto mais próxima do problema original ela está, melhor tende a ser o limitante inferior (para o caso de minimização) que a mesma fornece. A habilidade em produzir bons limitantes inferiores é essencial para o sucesso de um algoritmo de BBE, pois permite o descarte de subporções do espaço não promissoras de modo mais prematuro, evitando assim o desperdício de esforço em partes do espaço que não nos interessam.

Conforme discutido acima, uma vez que esta metodologia envolve a construção de uma relaxação convexa para o problema abordado, uma dificuldade marcante e evidente nas abordagens gerais baseadas em BBE consiste em como lidar, no momento da construção dessas relaxações, com os mais variados tipos de funções não convexas que podem estar presentes na função objetivo e/ou restrições. Em muitos casos, para se garantir a obtenção de uma relaxação convexa geral, pode-se terminar por construir uma relaxação fraca que tende a fornecer limitantes inferiores ruins e a prolongar o processo de convergência do algoritmo. Por esta razão, pode ser de grande interesse o desenvolvimento de abordagens especializadas para determinados tipos específicos de problema de otimização global, pois dessa forma, particularidades do problema em questão podem ser consideradas para a construção de uma relaxação convexa mais forte, levando assim a resultados melhores em relação à abordagens mais genéricas. Nesse trabalho, em especial, nos dispomos a desenvolver uma abordagem baseada em BBE para os problemas onde a não convexidade se remete apenas a termos quadráticos.

Para este trabalho, particularmente, importantes ideias foram obtidas a partir dos trabalhos [62] e [63], onde não convexidade quadrática é identificada dinamicamente e combinada com uma metodologia disjuntiva de modo bastante sofisticado. Nesta parte do nosso trabalho, no lugar de adotar um processo disjuntivo, desenvolvemos uma metodologia baseada em *Branch-And-Bound* Espacial para resolver o problema (PQR). Nossa abordagem se baseia em tirar o máximo de proveito possível de toda convexidade presente no problema, tanto nas funções convexas $f_j(x)$ quanto na parte quadrática $q_j(x)$, para a construção de uma boa relaxação convexa. A ideia principal consiste em decompor cada função quadrática $q_j(x) = \frac{1}{2}x'Q_jx$ como sendo uma diferença entre funções quadráticas convexas da seguinte forma:

$$q_j(x) = p_j(x) - r_j(x) \ , \tag{6.2}$$

onde $p_j(x) = \frac{1}{2}x'P_jx$ e $r_j(x) = \frac{1}{2}x'R_jx$ (note que as matrizes P_j e R_j são semidefini-

das positivas, já que $p_j(x)$ e $r_j(x)$ são convexas, e precisam ser calculadas de algum modo para que a decomposição seja obtida com sucesso). Por trás da decomposição (6.2), está a ideia de identificar uma possível parte convexa em $q_j(x)$ e separá-la no termo $p_j(x)$. Essa possibilidade de extração do termo $p_j(x)$ a partir de $q_j(x)$ é uma das principais vantagens de nossa abordagem. Por se tratar de um termo convexo, $p_j(x)$ não necessita ser relaxado. Assim, a decomposição (6.2) nos permite tratar na relaxação os termos convexos $p_j(x)$, juntamente com os termos $f_j(x)$, em suas formas originais, concentrando então nossos esforços na relaxação dos termos côncavos $-r_j(x)$ apenas. Note que essa estratégia tende a nos fornecer meios de construir uma relaxação melhor para (PQR) do que a que teríamos se simplesmente relaxássemos o termo $q_j(x)$ sem qualquer tentativa de identificação de uma parte convexa presente no mesmo. Posto isso, avaliamos aqui diferentes formas de se obter a decomposição (6.2), sendo umas delas baseada na decomposição por autovalores da matriz Q_j e, as demais, baseadas na decomposição dessa mesma matriz por meio da obtenção de uma matriz R_j diagonal que atenda aos requisitos de (6.2).

A estratégia de decompor funções como a diferença entre duas funções convexas no contexto de programação matemática é denominada como Diferença de Convexas (DC). DC já fundamentou diversos resultados teóricos e métodos de solução (para um apanhado geral sobre o assunto, consulte [64, 65]). Em particular, decomposições de funções quadráticas não convexas em uma diferença de funções convexas têm sido utilizadas na literatura para a construção de relaxações convexas dos problemas quadráticos abordados.

Em [66], um algoritmo de *Branch-And-Bound* elipsoidal foi proposto para minimizar um função quadrática não convexa sobre um conjunto convexo poliédrico limitado. Em [67], é introduzido o conceito de decomposições DC dominadas para funções quadráticas não convexas. Os autores demonstram que existe um número infinito de decomposições não dominadas para funções quadráticas não convexas (incluindo a decomposição por autovalores utilizada neste trabalho). Algoritmos para computar decomposições não dominadas são propostos pelos autores, mas os mesmos não apresentam resultados numéricos. Todavia, os autores mostram como o conceito pode ser aplicado para a computação de limitantes em algoritmos de BBE para problemas quadráticos. É demonstrado que limitantes obtidos com a subestimação por secantes da parte côncava de uma dada decomposição (para problemas de minimização) são mais fortes que limitantes obtidos com outras decomposições dominadas por essa primeira.

Em [68], é tratado o problema de minimização de uma função objetivo quadrática $x'Qx$, $x \in \mathbb{R}^n$ sobre um simplex. Os autores comparam dois limitantes inferiores para o valor da função objetivo: O limitante composto [69] e o limitante obtido por meio da DC $Q = P - R$. As matrizes semidefinidas positivas P e R são obtidas

por meio da resolução de um problema de programação semidefinida que garante o melhor limitante inferior entre qualquer escolha de matrizes. Embora os autores discutam sobre a extensão do limitante DC para o caso onde a região viável é dada por um politopo (ainda limitado), eles mencionam que, neste caso, o problema de programação semidefinida necessário para a obtenção de P e R é computacionalmente intratável. Adicionalmente, o esquema de decomposição DC proposto não é aplicável a problemas quadráticos mais gerais.

Um problema quadrático mais geral que os tratados nos trabalhos citados até aqui é tratado em [70], onde se minimiza a função quadrática não convexa $x'Q_0x + c_0'x$, $x \in \mathbb{R}^n$ sujeito a restrições quadráticas convexas $x'Q_ix + c_i'x + b_i \leq 0$, para $i = 1, \ldots, m$, ($Q_i \succeq 0$, isto é, Q_i é semidefinida positiva). Os autores propõem o cálculo de limitantes inferiores através da solução de relaxações convexas do problema. Estas relaxações convexas são construídas usando esquemas de decomposição DC otimizados. O termo côncavo é então subestimado por uma função linear sobre a relaxação. O esquema DC apresentado utiliza uma combinação de matrizes de posto 1 juntamente com as matrizes de restrições da seguinte forma:

$$Q_0 + \sum_{i=1}^{p} \lambda_i v_i v_i' + \sum_{i=1}^{m} \mu_i Q_i \succeq 0, \tag{6.3}$$

onde $v_i \in \mathbb{R}^n$, $i = 1, \ldots, p$, são vetores não nulos, $\lambda \in \mathbb{R}^p_+$ e $\mu \in \mathbb{R}^m_+$. Note então, que, para a obtenção de uma decomposição DC na forma $Q_0 = P_0 - R_0$, basta tomar $R_0 = \sum_{i=1}^{p} \lambda_i v_i v_i' + \sum_{i=1}^{m} \mu_i Q_i$ e então calcular P_0 como sendo $Q_0 + R_0$ (observe então que o lado esquerdo da expressão (6.3) fornece o valor para P_0 nesse contexto). Assim, de modo similar ao trabalho feito em [68], os autores utilizam um modelo de programação semidefinida para encontrar os valores de λ e μ que fornecem a combinação de matrizes segundo a expressão (6.3) que leva ao melhor limitante possível.

Nesta parte do trabalho, também consideramos decomposições DC de funções quadráticas não convexas presentes no problema abordado. Entretanto, o problema tratado aqui é mais geral que os problemas quadráticos abordados nos trabalhos citados. Ademais, diferentemente de [68, 70], nosso objetivo final não é apenas a construção de uma relaxação convexa para o cálculo de limitantes inferiores para o valor ótimo, mas sim a obtenção da solução global do problema abordado por meio de um algoritmo baseado em BBE. O Capítulo 7 apresenta os fundamentos de nossa abordagem, inicialmente para o caso particular do problema (PQR) onde todas as variáveis estão em domínio contínuo e a parte não convexa do problema está restrita à função objetivo, isto é, ao termo $q_0(x)$. Introduzimos nesse capítulo o algoritmo básico juntamente com distintas formas de se calcular a decomposição DC de $q_0(x)$. Posteriormente, propomos um modelo de programação semidefinida para a decisão

da melhor decomposição DC a ser adotada no problema abordado. Adicionalmente, discutimos ainda nesse capítulo a extensão de nossa metodologia para o caso mais geral representado no problema (PQR), onde não convexidades também podem estar presentes nas restrições através dos termos $q_i(x),\ i = 1, \ldots, m$, e com a possível presença de variáveis em domínios inteiros. Por fim, resultados computacionais com algumas conclusões são apresentados no Capítulo 8, onde também apresentamos outra grande contribuição desse trabalho de pesquisa, que consiste na disponibilização de um novo pacote computacional aberto que implementa a metodologia aqui descrita para a resolução de (PQR), o *solver* `iquad`.

Capítulo 7

Uma Abordagem de *Branch-And-Bound* Espacial para Programação Quadrática Não Convexa

7.1 Introdução

Para melhor apresentar os principais conceitos de nossa abordagem, tratamos neste capítulo da resolução de um caso particular do problema (PQR) apresentado no Capítulo 6, onde a não convexidade aparece apenas nos termos quadráticos presentes na função objetivo:

$$\begin{array}{cccc} (PQ) & \min\limits_{x} & f_0(x) \ + \ q_0(x), \\ & \text{sujeito a:} & x \in \mathcal{X} \end{array} \tag{7.1}$$

onde $f_0(x)$ é uma função convexa qualquer, possivelmente linear, não linear ou até mesmo nula, $q_0(x) = \frac{1}{2}x'Q_0x$ é uma função quadrática não convexa, o que significa dizer que Q_0 não é semidefinida positiva (assumimos Q_0 simétrica, todavia). $\mathcal{X}$ é um conjunto definido por restrições de caixa juntamente com uma quantidade m de restrições convexas:

$$\begin{array}{c} f_i(x) \leq b_i, \quad i = 1, \ldots, m \\ l_x \leq x \leq u_x, \\ x \in \mathbb{R}^n, \end{array} \tag{7.2}$$

onde $f_i(x),\ i = 1, \ldots, m$ são funções convexas quaisquer, possivelmente lineares, quadráticas ou não lineares. Assumimos aqui que (PQ) possui solução ótima global e que sua região viável é limitada. Nossa abordagem consiste na resolução do problema (PQ) por meio de um algoritmo baseado em *Branch-And-Bound* Espacial (BBE) que

resolve sucessivas relaxações convexas do problema (PQ) em subporções do espaço. Observe que, a característica das funções $f_j(x),\ j = 1, \ldots, m$ definirá o tipo de rotina computacional utilizada na resolução dessas relaxações. Se todas essas funções forem lineares, por exemplo, poderá ser empregado um *solver* de Programação Quadrática. Caso alguma função $f_i(x),\ i = 1, \ldots, m$ seja quadrática (sem que haja qualquer outra função não linear), poderá ser utilizado um *solver* para Problemas Quadraticamente Restritos. Caso haja alguma função não linear em (PQ), então poderá ser adotado um *solver* de Programação Não Linear.

Na Seção 7.8, discutimos a extensão da metodologia para problemas onde funções quadráticas não convexas também aparecem nas restrições dos problemas, além de uma possível incorporação nos mesmos de variáveis em domínios discretos (inteiros). Antes de especificamente introduzir os detalhes de nossa abordagem, é apropriado realizar uma breve discussão sobre BBE.

7.2 *Branch-And-Bound* Espacial

Apresentamos aqui o algoritmo de *Branch-And-Bound* Espacial (BBE) tomado por base para a construção de nossa metodologia (Algoritmo 7.1). A filosofia geral deste algoritmo é bastante similar ao *Branch-And-Bound* utilizado em Programação Inteira, apresentado neste trabalho na Seção 4.3 (Capítulo 4). Seja (P^{NC}) o problema de minimização não convexo sendo abordado. De modo geral, é necessário algum esquema para a construção de uma relaxação convexa desse problema, denotada aqui por $(\underline{\mathrm{P}}^{NC})$. Através de sua resolução, esta relaxação convexa é responsável pelo fornecimento de limitantes inferiores para diferentes subporções $\mathcal{V}^k$ do espaço da região viável do problema. Seja $(\underline{\mathrm{P}}^{NC}_{\mathcal{V}^k})$ a relaxação convexa de (P^{NC}) resolvida na subporção do espaço $\mathcal{V}^k$. Sejam ainda $\underline{x}^k$ a solução ótima de $(\underline{\mathrm{P}}^{NC}_{\mathcal{V}^k})$, caso este seja viável, $\underline{z}^k$ o seu respectivo valor ótimo da função objetivo, e, z^u, o melhor limitante superior conhecido para (P^{NC}) (note que z^u pode ser inicializado como $+\infty$). Note que se $\underline{x}^k$ for viável para (P^{NC}), é possível que esta solução possa atualizar o limitante superior z^u caso a mesma apresente valor objetivo em (P^{NC}) inferior a z^u. Assim, após essa possível etapa de atualização de z^u no BBE, temos três casos a considerar:

1. *O problema* $(\underline{P}^{NC}_{\mathcal{V}^k})$ *é inviável*: nesse caso, podemos descartar todos os subproblemas deste (subárvore abaixo do nó atual), pois também serão inviáveis (poda por inviabilidade, linha 16 do Algoritmo 7.1).

2. $\underline{z}^k \geq z^u$: neste caso, uma vez que o valor $\underline{z}^k$ é um limitante inferior para (P^{NC}) em toda subporção de $\mathcal{V}^k$, temos a certeza de que nenhuma das subporções

Entrada: (P^{NC}): Problema não convexo abordado, ϵ_c: tolerância de convergência
Saída: (x^*): solução ótima de (P^{NC})

$z^u = +\infty$;
Seja $f(.)$ a função objetivo de (P^{NC});
Seja ($\underline{\mathrm{P}}^{NC}$) uma relaxação convexa para (P^{NC});
Seja x o vetor de variáveis de decisão de ($\underline{\mathrm{P}}^{NC}$) ;
Seja $\mathcal{V}$ o conjunto (espaço) de soluções viáveis de ($\underline{\mathrm{P}}^{NC}$) ;
$\mathcal{V}^0 = \mathcal{V}$;
Seja ($\underline{\mathrm{P}}^{NC}_{\mathcal{V}^k}$) a relaxação convexa de (P^{NC}) na subporção do espaço $\mathcal{V}^k$;
Seja $N = \{0\}$ a lista inicial de nós em aberto;
$i = 0$;
Seja L^i o limitante inferior do nó i ;
$L^0 = -\infty$;
enquanto $N \neq \emptyset$ **faça**
    $z^l = min_j\{L^j \ : \ j \in N\}$;
    Selecione um nó k de N;
    se ($\underline{P}^{NC}_{\mathcal{V}^k}$) *é inviável* **então**
        $N = N\backslash\{k\}$; `// Poda por inviabilidade`
    senão
        Sejam $\underline{x}^k$ uma solução ótima de ($\underline{\mathrm{P}}^{NC}_{\mathcal{V}^k}$), e $\underline{z}^k$ seu respectivo valor na função objetivo deste problema;
        se $\underline{x}^k$ *é viável para* (P^{NC}) **então**
            $z^u = f(\underline{x}^k)$;
            $x^* = \underline{x}^k$;
            $N = N\backslash\{j : L^j \geq z^u - \epsilon_c\}$; `// Podas por limite`
        se $\underline{z}^k < z^u - \epsilon_c$ **então**
            `// Ramificação`
            Selecione uma variável x_j de ($\underline{\mathrm{P}}^{NC}_{\mathcal{V}^k}$) para ramificação ;
            Seja l^k_j e u^k_j, respectivamente, os limitantes inferior e superior de x_j em $\mathcal{V}^k$;
            Escolha $\psi \in (l^k_j \ \ u^k_j)$;
            $\mathcal{V}^{i+1} = \mathcal{V}^k \cap \{x : x_j \leq \psi\}$;
            $\mathcal{V}^{i+2} = \mathcal{V}^k \cap \{x : x_j \geq \psi\}$;
            $L^{i+1} = L^{i+2} = \underline{z}^k$;
            $N = N \cup \{i+1, i+2\}\backslash\{k\}$;
            $i = i + 2$;
        senão
            $N = N\backslash\{k\}$; `// Poda por limite`

Algoritmo 7.1: Algoritmo de *Branch-And-Bound Espacial* tomado por base em nossa abordagem.

desta contém qualquer solução que melhore o limitante superior conhecido, e, portanto, podemos descartá-las (poda por limite, linha 33 do Algoritmo 7.1).

3. $\underline{z}^k < z^u$: Neste caso, precisamos avaliar as subporções de $\mathcal{V}^k$, pois existe a possibilidade de ao menos alguma delas abrigar solução que melhore o limitante superior. Escolhemos uma variável x_j de ($\underline{\mathrm{P}}^{NC}$) para dividir o espaço. Sejam l_j^k e u_j^k, respectivamente, os limitantes inferior e superior de x_j na subporção do espaço $\mathcal{V}^k$ sendo correntemente explorada. Escolhemos um valor $\psi \in (l_j^k \;\; u_j^k)$ e, então, geramos duas novas subporções do espaço (ramificação, linhas 24-31 do Algoritmo 7.1). Na primeira, adicionamos a restrição $l_j^k \leq x_j \leq \psi$, e, na segunda, a restrição $\psi \leq x_j \leq u_j^k$. Note que, diferentemente do BB usado em programação inteira, as subporções geradas possuem interseção entre si, razão pela qual preferimos aqui usar o termo "divisão do espaço" em detrimento do termo "partição do espaço".

Ressaltamos que a poda por otimalidade aqui ocorre apenas quando a relaxação sendo resolvida fornece $\underline{x}^k$ viável com $\underline{z}^k$ (limitante inferior na subporção $\mathcal{V}^k$) tendo o mesmo valor objetivo de (P^{NC}) em $\underline{x}^k$ (limitante superior na subporção $\mathcal{V}^k$). Quando essa situação ocorre, essa poda por otimalidade acaba sendo executada como poda por limite no caso 2 acima. Apontamos que se o problema relaxado ($\underline{\mathrm{P}}^{NC}$) incorporar todas as restrições de (P^{NC}), o que só é possível se todas estas restrições forem convexas, cada solução $\underline{x}^k$ obtida nas subporções do espaço sempre será viável para (P^{NC}), embora não obrigatoriamente cada uma dessas soluções atualize o limitante superior z^u. Observe ainda que, no BB padrão de programação inteira, sempre que encontramos uma solução viável por meio da resolução da relaxação contínua em uma subporção do espaço não podada por limite, necessariamente atualizamos o limitante superior com essa nova solução, pois, uma vez que a respectiva subporção do espaço não foi podada por limite e a função objetivo da relaxação contínua é a mesma do problema de programação inteira abordado, essa solução ótima da relaxação contínua apresenta valor objetivo menor que o limitante superior corrente. A razão pela qual esse comportamento não necessariamente se repete no BBE é que a função objetivo de ($\underline{\mathrm{P}}^{NC}$) pode não ser a mesma de (P^{NC}), e, nesse caso, a relaxação convexa em uma subporção do espaço não podada por limite pode fornecer uma solução cujo valor objetivo em ($\underline{\mathrm{P}}^{NC}$) está abaixo de z^u, mas o valor objetivo dessa mesma solução em (P^{NC}) está acima de z^u.

Sempre que o limitante superior z^u for atualizado, pode-se descartar todas as subporções do espaço com limitante inferior igual ou maior que este valor (linha 22). Destacamos também que, no algoritmo BBE básico (Algoritmo 7.1), a região viável da relaxação convexa ($\underline{\mathrm{P}}^{NC}$) é que define a porção inicial do espaço (linha 5), já que esta região viável pode ser diferente (mais abrangente) da de (P^{NC}).

Destacamos também que a divisão do espaço é realizada considerando as variáveis de ($\underline{\mathrm{P}}^{NC}$) (linhas 24-31), uma vez que este último problema pode possuir variáveis auxiliares não originalmente presentes em (P^{NC}).

7.3 Construindo uma relaxação convexa

O primeiro passo para a aplicação de um algoritmo de BBE em um problema não convexo é a construção de uma relaxação convexa para o mesmo. Observe que o problema (PQ), em especial, possui região viável convexa, o que nos permite adotar todas as suas restrições em uma relaxação convexa sem maiores preocupações. Uma vez que a função $f_0(x)$ também é convexa, podemos voltar nossas atenções para a construção de um termo subestimador para a parte não convexa da função objetivo $q_0(x) = \frac{1}{2}x'Q_0x$. Para a realização desta tarefa, precisamos calcular matrizes simétricas semidefinidas positivas P_0 e R_0 com o objetivo de construir uma decomposição DC na forma $q_0(x) = p_0(x) - r_0(x)$, onde $p_0(x) = \frac{1}{2}x'P_0x$ e $r_0(x) = \frac{1}{2}x'R_0x$. Em outras palavras, precisamos inicialmente decompor Q_0 na forma $Q_0 = P_0 - R_0$.

7.3.1 Decomposição por Autovalores

Uma maneira natural de decompor a matriz Q_0, é por meio de sua decomposição por autovalores. Seja $\lambda_i^{Q_0}, i = 1,\ldots,n$ os autovalores de Q_0, com seus respectivos autovetores representados por $u_i, i = 1,\ldots,n$. Seja $\mathcal{A}^+_{Q_0}$ o conjunto de índices dos autovalores positivos de Q_0, isto é, $\mathcal{A}^+_{Q_0} = \{i \ : \ \lambda_i^{Q_0} > 0\}$, e $\mathcal{A}^-_{Q_0}$ o conjunto de índices dos autovalores negativos de Q_0, isto é $\mathcal{A}^-_{Q_0} = \{i \ : \ \lambda_i^{Q_0} < 0\}$. Deste modo, podemos calcular P_0 e R_0 como:

$$\begin{aligned} P_0 &= \sum_{i\in\mathcal{A}^+_{Q_0}} \lambda_i^{Q_0} u_i u_i' \ , \\ R_0 &= \sum_{i\in\mathcal{A}^-_{Q_0}} -\lambda_i^{Q_0} u_i u_i' \ . \end{aligned} \tag{7.3}$$

Uma vez que a decomposição (7.3) gera, de modo geral, matrizes R_0 cheias (não diagonais), a classificamos aqui como decomposição cheia, em oposição às decomposições alternativas apresentadas na Seção 7.5 que geram matrizes R_0 diagonais.

7.3.2 Nossa relaxação convexa

Uma vez feita, por algum modo, a decomposição $Q_0 = P_0 - R_0$, com as matrizes P_0 e R_0 simétricas semidefinidas positivas, estamos aptos a prosseguir na construção de uma relaxação convexa para (PQ). A partir de agora, precisamos escrever a expressão $\frac{1}{2}x'R_0x$ em uma forma separável. Para isso, tomamos a decomposição

por autovalores de R_0. Sejam λ_i os autovalores de R_0, e v_i os seus respectivos autovetores. Note que R_0 não possui autovalor negativo. Seja $\mathcal{A}^+$ o conjunto de índices de autovalores positivos (não nulos) de R_0, isto é, $\mathcal{A}^+ = \{i \ : \ \lambda_i > 0\}$. Escrevemos então R_0 como:

$$R_0 = \sum_{i \in \mathcal{A}^+} \lambda_i v_i v_i' \ .$$

Introduzindo as variáveis auxiliares

$$y_i := v_i' x, \ i \in \mathcal{A}^+ \ , \tag{7.4}$$

reformulamos o problema (PQ) como:

$$\begin{array}{lllll} (\bar{PQ}) & \min\limits_{x,\,y} & f_0(x) \ + \frac{1}{2} x' P_0 x \ - \ \frac{1}{2} \sum\limits_{i \in \mathcal{A}^+} \lambda_i y_i^2 & & \\ & \text{sujeito a:} & x \in \mathcal{X}, & & \\ & & y_i = v_i' x, & i \in \mathcal{A}^+, & \\ & & l_{y_i} \leq y_i \leq u_{y_i}, & i \in \mathcal{A}^+, & \\ & & y \ \in \ \mathbb{R}^{|\mathcal{A}^+|}, & & \end{array} \tag{7.5}$$

onde os limitantes l_{y_i} e u_{y_i} podem ser calculados, por exemplo, por meio da resolução dos seguintes problemas auxiliares, para $i \in \mathcal{A}^+$:

$$\begin{array}{llll} (L_i^y) & l_{y_i} := & \min\limits_{x} & v_i' x, \\ & & \text{sujeito a:} & x \in \mathcal{X}, \end{array} \tag{7.6}$$

e

$$\begin{array}{llll} (U_i^y) & u_{y_i} := & \max\limits_{x} & v_i' x, \\ & & \text{sujeito a:} & x \in \mathcal{X}, \end{array} \tag{7.7}$$

usando um *solver* apropriado em relação às restrições que definem $\mathcal{X}$. Note que, para acelerar o processo de obtenção dos limitantes l_{y_i} e u_{y_i}, é possível relaxar algumas restrições que definem $\mathcal{X}$ nos problemas (L_i^y) e (U_i^y) (desde que suas regiões viáveis permaneçam limitadas). Também é possível não resolver estes problemas até a otimalidade utilizando então seus respectivos limitantes duais obtidos como valores para l_{y_i} e u_{y_i}. De todo modo, ter em mãos limitantes os mais justos possíveis para as variáveis pode ser bastante benéfico para acelerar a convergência de um algoritmo baseado em BBE.

Tendo apresentado a reformulação $(\bar{PQ})$, podemos considerar uma relaxação para este problema. É válido destacar que função objetivo de $(\bar{PQ})$ é composta do termo convexo $f_0(x) + \frac{1}{2} x' P_0 x$ juntamente com termos côncavos em função de y.

Note que, cada um desses termos côncavos tem a forma:

$$\begin{array}{c} \omega_i(y_i) := -\frac{1}{2}\lambda_i y_i^2, \\ l_{y_i} \leq y_i \leq u_{y_i}, \end{array} \tag{7.8}$$

para $i \in \mathcal{A}^+$. Podemos então subestimar esses termos côncavos através de secantes. Substituímos cada termo $-y_i^2$ por uma nova variável w_i, a qual submetemos à desigualdade secante:

$$-(u_{y_i} + l_{y_i})y_i + l_{y_i}u_{y_i} \leq w_i. \tag{7.9}$$

Assim, construímos a nossa almejada relaxação convexa:

$$\begin{array}{lll} (\underline{\tilde{PQ}}) \quad \min\limits_{x,\,y,\,w} & f_0(x) + \frac{1}{2}x'P_0x + \frac{1}{2}\sum\limits_{i\in\mathcal{A}^+}\lambda_i w_i & \\ \text{sujeito a:} & x \in \mathcal{X}, & \\ & y_i = v_i'x, & i \in \mathcal{A}^+, \\ & -(u_{y_i} + l_{y_i})y_i + l_{y_i}u_{y_i} \leq w_i, & i \in \mathcal{A}^+, \\ & l_{y_i} \leq y_i \leq u_{y_i}, & i \in \mathcal{A}^+, \\ & y, w \in \mathbb{R}^{|\mathcal{A}^+|}. & \end{array} \tag{7.10}$$

Para facilitar o processo de resolução, pode ser proveitoso omitir as variáveis y_i, obtendo assim:

$$\begin{array}{lll} (\underline{PQ}) \quad \min\limits_{x,\,w} & f_0(x) + \frac{1}{2}x'P_0x + \frac{1}{2}\sum\limits_{i\in\mathcal{A}^+}\lambda_i w_i & \\ \text{sujeito a:} & x \in \mathcal{X}, & \\ & -(u_{y_i} + l_{y_i})v_i'x + l_{y_i}u_{y_i} \leq w_i, & i \in \mathcal{A}^+, \\ & l_{y_i} \leq v_i'x \leq u_{y_i}, & i \in \mathcal{A}^+, \\ & w \in \mathbb{R}^{|\mathcal{A}^+|}. & \end{array} \tag{7.11}$$

Sejam $l_{y_i}^k$ e $u_{y_i}^k$ os limitantes inferiores e superiores, respectivamente, das varáveis y_i, $i \in \mathcal{A}^+$, na k-ésima subporção do espaço. Assim o subproblema ($\underline{\text{PQ}}^k$) resolvido nessa subporção é dado por:

$$\begin{array}{lll} (\underline{\text{PQ}}^k) \quad \min\limits_{x,\,w} & f_0(x) + \frac{1}{2}x'P_0x + \frac{1}{2}\sum\limits_{i\in\mathcal{A}^+}\lambda_i w_i & \\ \text{sujeito a:} & x \in \mathcal{X}, & \\ & -(u_{y_i}^k + l_{y_i}^k)v_i'x + l_{y_i}^k u_{y_i}^k \leq w_i, & i \in \mathcal{A}^+, \\ & l_{y_i}^k \leq v_i'x \leq u_{y_i}^k, & i \in \mathcal{A}^+, \\ & w \in \mathbb{R}^{|\mathcal{A}^+|}. & \end{array} \tag{7.12}$$

Note que as desigualdades secantes (7.9) presentes em ($\underline{\text{PQ}}^k$) são atualizadas pelos valores de $l_{y_i}^k$ e $u_{y_i}^k$ da k-ésima subporção do espaço k.

7.4 Pormenores do Algoritmo de *Branch-And-Bound* Espacial

Durante a aplicação de nossa abordagem baseada em BBE, a divisão do espaço é feita através das variáveis y_i. No momento apropriado para ramificação, é necessária a escolha de um índice $j \in \mathcal{A}^+$ para então procedermos à divisão do espaço sobre y_j. Um valor $\psi \in (l^k_{y_j}\ u^k_{y_j})$ é tomado e então duas novas subporções do espaço são geradas. Na primeira delas, impomos a restrição $l^k_{y_j} \leq y_j \leq \psi$, enquanto que na outra, impomos a restrição $\psi \leq y_j \leq u^k_{y_j}$. Observe que, assim, as desigualdades secantes são construídas em um intervalo menor a cada nova divisão do espaço, o que tende a melhorar a aproximação dada pelas mesmas para os termos $-y_i^2$.

Seja ($\underline{x}^k$,$\underline{w}^k$) uma solução ótima de ($\underline{\text{PQ}}^k$) (quando este problema é viável). Para a escolha do índice j para a ramificação, podemos tomar aquele com a maior diferença entre $-\lambda_i(v_i'\underline{x}^k)^2$ e $\lambda_i\underline{w}^k_i$, $i \in \mathcal{A}^+$ (maior discrepância entre o valor do termo côncavo na função objetivo e sua aproximação). Uma outra possível escolha é o índice que apresente o maior valor $\lambda_i(u^k_{y_i} - l^k_{y_i})$. Para a escolha de ψ (o valor sobre o qual a divisão do espaço será feita na caixa de y_j), pode se tomar $\psi = v_j\text{'}\underline{x}^k$. Entretanto, esta pode ser uma escolha ruim se esse valor estiver demasiadamente próximo dos limitantes do intervalo $[l^k_{y_j}\ u^k_{y_j}]$. Alternativamente, pode-se o escolher o ponto central deste intervalo, isto é $\psi = \frac{1}{2}(u^k_{y_j} + l^k_{y_j})$, ou ainda, uma combinação de ambas as estratégias através de um parâmetro $\alpha \in [0\ 1]$ de acordo com a seguinte expressão:

$$\psi = \alpha v_j'\underline{x}^k + (1-\alpha)\frac{(u^k_{y_j} + l^k_{y_j})}{2} \quad . \tag{7.13}$$

Com relação à escolha da próxima subporção do espaço a ser explorada, podem ser adotadas as já conhecidas estratégias de exploração por limitante inferior mais baixo, profundidade ou largura.

7.5 Decomposições Diagonais

Apresentamos nessa seção as decomposições diagonais como uma alternativa à decomposição por autovalores, apresentada na Subseção 7.3.1, para decompor Q_0 na forma $Q_0 = P_0 - R_0$. Denominamos estas decomposições como diagonais porque elas geram matrizes R_0 diagonais. Uma vantagem em se trabalhar com matrizes R_0 diagonais, é que seus autovetores v_i são colunas e_j da matriz identidade, isto é vetores preenchidos com um na posição j e zero nas demais posições. Desse modo, de acordo com 7.4, cada variável y_i do problema relaxado ($\underline{\tilde{\text{PQ}}}$) estará diretamente relacionada à uma variável x_j (terão o mesmo valor). Assim, os limitantes l_{y_i} e u_{y_i} podem ser tomados como os próprios limitantes de x_j, l_{x_j} e u_{x_j}. Note então que,

ao dividir o espaço sobre algum y_i divide-se também diretamente o espaço de x_j, e se, por ventura, a variável x_j for restrita a domínio discreto (inteiro), a variável y_i também ficará restrita a esse mesmo domínio. Nesse contexto, como realizar a ramificação sobre y_i significa realizar a ramificação sobre x_j, e, sendo esta última uma variável inteira, a ramificação do BB de programação inteira poderia ser adotada. Desta forma, com uma decomposição diagonal, as variáveis y_i podem ser completamente omitidas, sendo então cada y_i substituída respectivamente por sua variável relacionada x_j. Assim, as restrições de caixa em cada y_i também podem ser omitidas.

As decomposições diagonais também fornecem sempre a mesma base de vetores v_i para a construção da relaxação convexa, que são as colunas e_j da matriz identidade. Essa característica pode ser útil quando o problema possuir múltiplas matrizes Q_k para serem decompostas, como, por exemplo, no caso de existirem restrições quadráticas não convexas no problema abordado. Nesse, caso, com todas as matrizes R_k partilhando o mesmo conjunto de autovetores, o número de variáveis auxiliares y_i e w_i bem como o número de desigualdades secantes seria reduzido, facilitando assim a resolução da relaxação convexa. Paralelamente, ao realizar a ramificação sobre alguma variável y_i, melhoraríamos as aproximações secantes simultaneamente em todas as expressões (restrições e/ou função objetivo) que fizeram uso de y_i para compor a relaxação convexa.

Uma possível desvantagem das decomposições diagonais é que, uma matriz R_0 diagonal que satisfaça aos critérios da decomposição DC em questão pode, em alguns casos, apresentar valores demasiadamente grandes para seus autovalores λ_i, o que tende a acentuar a discrepância ente os termos côncavos $-\lambda_i y_i^2$ e seu respectivos aproximadores $\lambda_i w_i$. Essa possível discrepância alta pode exigir níveis maiores de divisão do espaço (ramificação) no algoritmo de BBE de modo que as desigualdades secantes façam os termos $\lambda_i w_i$ aproximarem adequadamente os termos $-\lambda_i y_i^2$. Uma outra possível desvantagem é que, uma vez que os autovetores de R_0 são os vetores e_j, que possuem apenas um elemento não nulo, há uma tendência de que as decomposições diagonais gerem uma matriz R_0 com uma quantidade maior de autovetores relacionados a autovalores positivos (não nulos) em relação às decomposições cheias, cujos autovetores podem possuir mais de um elemento não nulo. Em outras palavras, as decomposições cheias tendem a conseguir gerar uma matriz R_0 que possua um número menor de autovetores relacionados a autovalores positivos. Desse modo, as decomposições diagonais tendem a gerar um número maior de termos côncavos $-y_i^2$ para serem aproximados por secantes.

A seguir, apresentamos algumas decomposições diagonais para Q_0. Lembramos que, aqui, R_0 é uma matriz diagonal cujos autovetores são as colunas e_j da matriz identidade. Denotamos aqui $r_0 := (r_{01}, r_{02}, \ldots, r_{0n})$ como sendo o vetor formado

pelos elementos na diagonal de R_0. Note que este vetor traz também os autovalores de R_0. Uma vez tendo calculado R_0, pode-se então obter P_0 com a expressão $P_0 = Q_0 + R_0$.

7.5.1 Diagonalmente Dominante

Uma maneira bem simples de se obter uma matriz R_0 diagonal é utilizando o conceito de diagonal dominância. Calculamos então cada r_{0i} como:

$$r_{0i} := \max\left\{0,\ -q_{0ii} + \sum_{j:j\neq i} |q_{0ij}|\right\},$$

onde q_{0ij} denota o elemento na linha i e coluna j da matriz Q_0. Note que, $P_0 := Q_0 + R_0$ será uma matriz diagonal dominante. É fácil perceber que esta forma de decomposição pode gerar autovalores λ_i para R_0 demasiadamente grandes, o que pode ser uma desvantagem prática na aplicação do algoritmo BBE conforme discutido acima.

7.5.2 Identidade

Uma alternativa possivelmente melhor de se obter R_0 é por meio do uso do menor autovalor de Q_0, denotado aqui por $\underline{\lambda}^{Q_0}$. Assim, definimos R_0 como:

$$R_0 = -\min\{0,\ \underline{\lambda}^{Q_0}\}\ I \quad ,$$

onde I é a matriz identidade. Note que se Q_0 não for semidefinida positiva, $\underline{\lambda}^{Q_0}$ será negativo.

7.5.3 Diagonal PSD

Uma maneira mais robusta de se conseguir R_0 é através da resolução do seguinte Problema de Programação SemiDefinida (PSD):

$$\begin{aligned} (DDPSD) \qquad \min_{r} \quad & \sum_{i=1}^{n} r_i \ , \\ \text{sujeito a:} \quad & Q_0 + \text{Diag}(r) \succeq 0 \ , \\ & r \geq 0 \ , \\ & r \in \mathbb{R}^n, \end{aligned} \tag{7.14}$$

onde a expressão $M \succeq 0$ significa dizer que M é uma matriz semidefinida positiva, e o operador Diag(r) constrói uma matriz diagonal cujos elementos i,i na diagonal são definidos por r_i. Definimos então $R_0 := \text{Diag}(r)$. Assim, o modelo acima tem como objetivo minimizar o traço de R_0. De certa forma, é como se este modelo procurasse

minimizar a convexidade estritamente separável necessária para ser adicionada em Q_0 e obter como resultado uma matriz semidefinida positiva.

7.6 Decomposição por Minimização da Soma Ponderada de Autovalores

Nesta seção, definimos uma classe interessante de decomposições cheias (não diagonais) para uma matriz simétrica não semidefinida positiva Q, na forma $Q = P - R$, onde P e R são matrizes simétricas semidefinidas positivas. Consideramos decomposições na forma $q(x) = p(x) - r(x)$, onde $q(x) = \frac{1}{2}x'Qx$, $p(x) = \frac{1}{2}x'Px$ e $r(x) = \frac{1}{2}x'Rx$. Aqui, estamos interessados em Q, P e R que resolvem:

$$\begin{array}{lll} (PSPAV) & \min\limits_{R} & \sum\limits_{i=1}^{k} m_i\lambda_i(R) \ , \\ & \text{sujeito a:} & Q + R \succeq 0 \ , \\ & & R \succeq 0 \ , \end{array} \tag{7.15}$$

onde $m_1 \geq m_2 \geq \cdots m_k > 0$, k é um parâmetro. A função objetivo é composta pela norma 1 ponderada dos autovalores da matriz R. Desse modo, estamos de certa forma minimizando a não convexidade de $-r(x)$ objetivando um melhor desempenho da relaxação convexa em fornecer limitantes inferiores para os subproblemas abordados e, por consequência, melhorar a eficiência do algoritmo de BBE de um modo geral.

Observe que, se definirmos $k := n$ e $m_i := 1$, $i = 1,2,\ldots,n$, o problema $(PSPAV)$ recai na minimização do traço de R. Como este problema permite a obtenção de R cheia (não diagonal), espera-se que a decomposição obtida aqui seja melhor do que aquela obtida com a decomposição diagonal PSD apresentada na Subseção 7.5.3 no sentido de melhor extrair uma possível convexidade presente em Q. Em contrapartida, o custo computacional na resolução deste novo problema tende a ser maior em comparação com aquele da Subseção 7.5.3.

Alternativamente, definindo $k := 1$ e $m_1 := 1$, recaímos no problema de minimizar o maior autovalor de R. Nesse caso, o problema $(PSPAV)$ pode ser reformulado como:

$$\begin{array}{lll} \min\limits_{z,\,R} & z \ , \\ \text{sujeito a:} & zI - R \succeq 0 \ , \\ & Q + R \succeq 0 \ , \\ & R \succeq 0 \ . \end{array} \tag{7.16}$$

O problema $(PSPAV)$ define uma família de decomposições que podem nos

servir em nosso propósito de construir uma relaxação convexa para ser utilizada no BBE. Demonstraremos a seguir, entretanto, que todos esses problemas de minimização de soma ponderada de autovalores podem ser resolvidos de forma muito eficiente apenas por meio da decomposição por autovalores apresentada na Subseção 7.3.1. Este fato nos dá uma ideia do quão importante é a decomposição por autovalores, e é enunciado no Teorema a seguir:

Teorema 7.1. *Para quaisquer parâmetros m_i, $i = 1, \ldots, k$, tais que $m_1 \geq m_2 \geq \cdots m_k > 0$, a decomposição por autovalores de Q resolve o problema de minimização de soma ponderada de autovalores ($PSPAV$).*

A demonstração desse Teorema pode ser conferida no Apêndice B.

7.7 Uma estratégia de decomposição alternativa baseada em PSD

Apresentamos aqui uma diretriz geral para a obtenção de decomposições de uma matriz Q simétrica não semidefinida positiva na forma $Q = P - R$, onde P e R são matrizes simétricas semidefinidas positivas. A ideia principal consiste em tomar um conjunto de matrizes de posto um $\{V_i \,|\, V_i = v_i v_i'\}$ e calcular a melhor decomposição possível para Q segundo algum critério. Este cálculo é feito através da resolução do seguinte problema de Programação SemiDefinida (PSD):

$$
\begin{aligned}
(DQPSD) \qquad \min_{\lambda \in \mathbb{R}^p} \quad & \sum_{i=1}^{p} \beta_i \lambda_i \ , \\
\text{sujeito a:} \quad & Q \ + \ \sum_{i=1}^{p} \lambda_i V_i \ \succeq 0 \ , \\
& \lambda \geq 0 \ ,
\end{aligned}
\tag{7.17}
$$

onde $\beta \geq 0$ é um vetor em que cada componente β_i contém o peso para a variável λ_i na função objetivo. Para completar o processo de decomposição, tomamos $R = \sum_{i=1}^{p} \lambda_i V_i$ e $P = Q + R$. Observamos que a resolução do problema $(DQPSD)$ nos fornece uma matriz R que atende aos requisitos de nossa decomposição estando já decomposta sobre v_i e λ_i. Assim, nossa estratégia calcula a melhor matriz R possível de acordo com os parâmetros β_i e v_i, $i = 1, \ldots, p$. Podemos notar que este é o caminho reverso ao das decomposições apresentadas até aqui, que primeiramente obtém R de algum modo, e só então calculam os vetores v_i por meio de sua decomposição. Em outras palavras, as decomposições apresentadas anteriormente "escolhem" R e então calculam os vetores v_i em função deste. Esta nova estratégia primeiro escolhe os vetores v_i e então calcula R em função deles.

Diferentes escolhas para os vetores β e v_i conduzirão a formas variadas de decomposição. Por exemplo, tomando $\beta_i = 1$ e $v_i = e_i$, onde e_i é a i-ésima coluna da

matriz identidade, para $i = 1, \ldots, n$, o problema $(DQPSD)$ recai no problema:

$$
\begin{aligned}
(NDDPSD) \qquad \min_{\lambda \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} \lambda_i \ , \\
\text{sujeito a:} \quad & Q \ + \ \sum_{i=1}^{n} \lambda_i E_i \ \succeq 0 \ , \\
& \lambda \geq 0 \ ,
\end{aligned}
\tag{7.18}
$$

onde $E_i = e_i e_i'$. Note que $(NDDPSD)$ é equivalente ao problema $(DDPSD)$ que é utilizado para calcular R segundo a decomposição diagonal PSD em (7.14). Apontamos que a formulação acima traz um modelo prático eficiente para o cálculo de R segundo esta última decomposição.

Uma outra escolha para os parâmetros β e v_i que pode ser destacada, seria tomar $\beta_i = 1$ e cada v_i como sendo um dos autovetores de Q relacionados a algum autovalor negativo, para $i = 1,..,d$, onde d é o número de autovalores negativos de Q. Claramente, os autovalores negativos de Q (multiplicados por -1) fornecem uma solução para $(DQPSD)$ nesse contexto. Assim, a decomposição gerada seria a mesma obtida com a decomposição por autovalores. Esse fato poderia ser futuramente explorado, para, por exemplo, obter uma decomposição para Q o mais próximo possível da decomposição por autovalores sujeito a possíveis restrições adicionais em $(DQPSD)$.

7.7.1 Decidindo Entre Decomposição por Autovalores e Diagonal PSD

Temos observado que, na prática, as decomposições por autovalores e diagonal PSD duelam pelo melhor desempenho quando incorporadas em nosso algoritmo baseado em BBE. Temos observado, através de experimentos práticos, que não há predominância absoluta entre essas duas formas de decomposição, o que nos leva a acreditar que a melhor escolha esteja intimamente relacionada às características específicas do problema sendo tratado. Deste modo, torna-se importante desenvolver uma maneira funcional de decidir sobre qual forma de decomposição será aplicada no problema em questão, visando obter os melhores resultados possíveis por meio da aplicação de BBE.

Uma possível forma de realizar a escolha pela melhor decomposição, é construindo um problema de PSD a partir de $(DQPSD)$, onde tomamos como v_i os autovetores u_j relacionados aos autovalores negativos de Q juntamente com os vetores canônicos e_k utilizados pela decomposição por diagonal PSD em $(NDDPSD)$. Em outras palavras, mesclamos os vetores v_i obtidos com as decomposições de autovalores e diagonal PSD em um modelo unificado. Para tal, sejam $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_d < 0$ os d autovalores negativos de Q e u_i, $i = 1,2, \ldots, d$ seus respectivos autovetores. De-

finimos então, para $i = 1,2,\ldots,d+n$:

$$v_i := \begin{cases} u_i, & \text{se } i \leq d \ , \\ e_{i-d}, & \text{caso contrário} \ . \end{cases} \tag{7.19}$$

Aplicando (7.19) no modelo $(DQPSD)$ e definindo $U_i = u_i u_i'$ e $E_i = e_i e_i'$ temos:

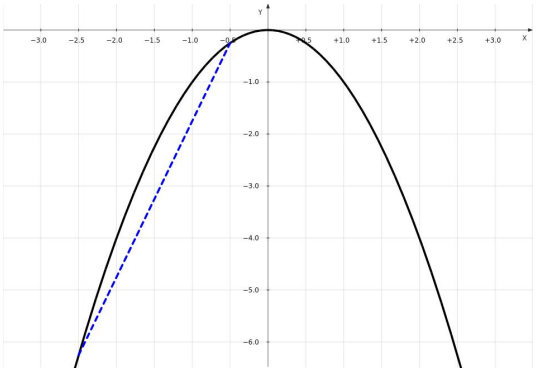
$$\begin{aligned} (MIX0) \quad \min_{\lambda \in \mathbb{R}^{d+n}} \quad & \sum_{i=1}^{d+n} \beta_i \lambda_i \ , \\ \text{sujeito a:} \quad & Q \ + \ \sum_{i=1}^{d} \lambda_i U_i \ + \ \sum_{i=1}^{n} \lambda_{d+i} E_i \ \succeq 0 \ , \\ & \lambda \geq 0 \ . \end{aligned} \tag{7.20}$$

Assim, a última coisa a fazer para completar a definição do modelo $(MIX0)$ é escolher valores apropriados para β na função objetivo. A escolha $\beta_i = 1$ se mostra inapropriada nesta situação, pois levaria o modelo a sempre preferir compor R com os autovetores u_i em vez dos vetores canônicos e_i, visto que, conforme enunciado pelo Teorema 7.1, a decomposição por autovalores nos dá a solução que minimiza o traço de R. Nosso desejo é que o modelo seja capaz de escolher, entre a decomposição por autovalores e a diagonal PSD, aquela que gerará a melhor relaxação para o problema não convexo abordado. É natural pensar que, além da matriz Q presente na função objetivo de (PQ), o modelo deve levar em consideração, de algum modo, as restrições desse problema. Como nesta metodologia já sabemos a priori os possíveis vetores sobre os quais R será decomposto, podemos calcular os limitantes inferiores e superiores para cada um deles na região viável $\mathcal{X}$ conforme feito nos problemas (7.6) e (7.7). Assim, para $i = 1,2,\ldots,d+n$, uma possível escolha para β_i seria:

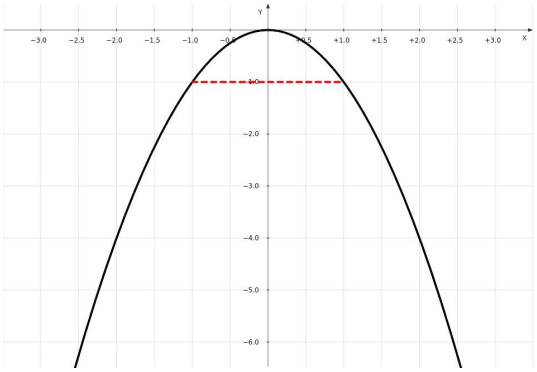
$$\beta_i = \hat{u}(v_i) - \hat{l}(v_i) \ ,$$

onde $\hat{l}(v_i) = \min\{v_i' x : x \in \mathcal{X}\}$ e $\hat{u}(v_i) = \max\{v_i' x : x \in \mathcal{X}\}$. A ideia aqui seria privilegiar direções v_i com menor comprimento na região viável $\mathcal{X}$. Desse modo, as desigualdades secantes seriam construídas em intervalos os menores possíveis, esperando-se assim diminuir a discrepância entre a aproximação secante e o real valor do termo côncavo aproximado.

Todavia, considerar apenas o comprimento do intervalo sendo aproximado pela secante pode não ser a melhor escolha para β_i, conforme evidenciado pela Figura 7.1. Nesta figura, temos a função $g(a) = -a^2$, a mesma usada em nossa reformulação para (PQ), sendo aproximada por desigualdades secantes em dois intervalos distintos tendo, cada um, comprimento dois. Claramente, a primeira secante, 7.1a, aproxima a função em seu respectivo intervalo melhor que a segunda, 7.1b, uma vez que, o segundo intervalo abriga a mudança no sinal da derivada em seu centro. Desse modo, podemos concluir que podem haver direções com largos intervalos e, ainda assim,



a aproximação secante sobre $a \in [-2.5, -0.5]$



b aproximação secante sobre $a \in [-1, 1]$

Figura 7.1: Aproximações secantes para $g(a) = -a^2$ sobre dois intervalos diferentes de comprimento dois. Note que, embora os comprimentos dos intervalos sejam iguais, em 7.1a temos uma aproximação consideravelmente melhor que em 7.1b.

capazes de prover boas aproximações secantes, ao mesmo tempo em que podem haver direções com intervalos mais curtos fornecendo aproximações secantes ruins.

Assim, adotamos uma estratégia ligeiramente diferente para definir β. Nossa metodologia geral para resolver (PQ) envolve a construção de aproximações secantes de:

$$g(a) = -a^2$$

nos intervalos $[\hat{l}_i \ \hat{u}_i]$, onde $\hat{l}_i = \hat{l}(v_i)$, $\hat{u}_i = \hat{u}(v_i)$ e v_i são os vetores utilizados no modelo $(MIX0)$. Nesse contexto, as funções secantes podem ser definidas como:

$$s_i(a) = -(\hat{u}_i + \hat{l}_i)a + \hat{u}_i\hat{l}_i \ .$$

Nossa intenção principal em relação ao modelo $(MIX0)$ é a obtenção de uma solução λ minimizando a soma da diferença máxima entre $g(a)$ e cada uma das funções secantes $s_i(a)$. Então, definimos cada β_i como:

$$\beta_i = \max \ \delta_i(a)$$

onde

$$\delta_i(a) := g(a) - s_i(a) = -a^2 + (\hat{u}_i + \hat{l}_i)a - \hat{u}_i\hat{l}_i \ .$$

Uma vez que as funções $\delta_i(a)$ são quadráticas, univariadas e côncavas, podemos determinar facilmente seus valores máximos como $\frac{\hat{u}_i^2+\hat{l}_i^2}{4} - \frac{\hat{u}_i\hat{l}_i}{2}$ no ponto $a = \frac{\hat{u}_i+\hat{l}_i}{2}$. Dessa forma, definimos $\beta_i = \frac{\hat{u}_i^2+\hat{l}_i^2}{4} - \frac{\hat{u}_i\hat{l}_i}{2}$ e o problema $(MIX0)$ se torna:

$$
\begin{array}{lll}
(MIX) & \min\limits_{\lambda \in \mathbb{R}^{d+n}} & \sum\limits_{i=1}^{d+n} \left(\frac{\hat{u}_i^2 + \hat{l}_i^2}{4} - \frac{\hat{u}_i \hat{l}_i}{2} \right) \lambda_i \ , \\
& \text{sujeito a:} & Q \ + \ \sum\limits_{i=1}^{d} \lambda_i U_i \ + \ \sum\limits_{i=1}^{n} \lambda_{d+i} E_i \ \succeq 0 \ , \\
& & \lambda \geq 0 \ .
\end{array}
\tag{7.21}
$$

Ressaltamos que a função objetivo de (MIX) nos dá uma medida da maior discrepância possível entre a soma das funções $g(a)$ e a soma das suas respectivas aproximações secantes $s_i(a)$ (multiplicadas pelos seus respectivos termos λ_i). Ao minimizar este valor, esperamos realizar a escolha adequada entre os autovetores u_i de Q e os vetores canônicos e_k para a composição de uma matriz R que resulte em uma boa relaxação para o problema (PQ), contribuindo assim para um bom desempenho por parte do algoritmo de BBE. Note que modelo (MIX) pode escolher usar apenas os autovetores u_i, ou apenas os vetores canônicos e_i ou ainda, uma combinação desses dois conjuntos de vetores.

7.8 Extensões da Metodologia

7.8.1 Considerando Funções Quadráticas Não Convexas Nas Restrições

Tendo apresentado os principais conceitos de nossa abordagem para problemas envolvendo uma função quadrática não convexa na função objetivo, estamos prontos para discutir sobre a extensão da abordagem para problemas onde termos quadráticos não convexos também podem aparecer nas suas restrições. Em outras palavras, nesta seção, passamos a abordar o problema:

$$
\begin{array}{lll}
(PQR) \quad z := & \min\limits_{x} & f_0(x) \ + \ q_0(x), \\
& \text{sujeito a:} & x \in \mathcal{X}, \\
& & h_j(x) \ + \ q_j(x) \leq c_j, \quad j = 1, \ldots, t \ ,
\end{array}
\tag{7.22}
$$

onde, $\mathcal{X}$ é ainda um conjunto definido por restrições convexas conforme (7.2), as funções $h_j(x),\ i = 1, \ldots, t$, assim como $f_0(x)$, são não lineares e convexas. Aqui, assumimos que todas as restrições convexas estão representadas em $\mathcal{X}$, restando então para $q_j(x) = \frac{1}{2} x' Q_j x,\ j = 1, \ldots, t$ a representação de funções quadráticas não convexas. Observe, todavia, que a função $q_0(x)$ na função objetivo pode ser nula.

O problema (PQR) admite a existência de múltiplas funções $q_j(x)$ não convexas, o que traz um desafio adicional para a construção de uma boa relaxação convexa.

Nossa metodologia geral apresentada aqui estabelece os seguintes passos a obtenção dessa relaxação:

1. Construção de uma decomposição para cada matriz $Q_j = P_j - R_j$ tal que P_j e R_j sejam semidefinidas positivas;

2. Decomposição por autovalores de cada matriz R_j para introdução das variáveis auxiliares conforme expressão (7.4);

3. Substituição dos termos quadráticos côncavos sobre as variáveis introduzidas no passo 2 por termos aproximadores submetidos a desigualdades secantes.

Observe que, em princípio, cada matriz R_j obtida poderia apresentar um conjunto distinto de autovetores, o que poderia acarretar na introdução de um número elevado de variáveis auxiliares e desigualdades secantes. Este fato, além de possivelmente deixar as relaxações contínuas mais dispendiosas de serem resolvidas, poderia exigir muitos níveis de ramificações no algoritmo de BBE para uma adequada aproximação do problema (PQR). Em um caso extremo, onde nenhum autovetor é partilhado por duas matrizes R_j quaisquer, a cada ramificação, a aproximação por secantes seria melhorada para apenas uma das funções quadráticas não convexas, e, ainda assim, em apenas um dos termos côncavos utilizados na reformulação da respectiva função, o que não seria ideal para o BBE em termos da agilidade de sua convergência.

Com estas informações em mente, podemos construir uma relaxação convexa onde toda matriz R_j seja construída a partir do mesmo conjunto de autovetores. Desse modo, o número de variáveis auxiliares e desigualdades seria reduzido, e, a cada ramificação, a aproximação por secantes poderia ser melhorada para diversas funções quadráticas simultaneamente, o que aceleraria a convergência do algoritmo de BBE. Conforme discutido na Seção 7.5, uma sugestão natural seria utilizar decomposições diagonais para todas as matrizes Q_j. Dessa forma, cada matriz R_j obtida teria como autovetores as colunas e_i da matriz identidade, o que atenderia nosso propósito de construir cada R_j sobre um mesmo conjunto de autovetores. Entretanto, existem muitos casos onde mesmo a decomposição diagonal PSD, que domina as demais decomposições diagonais apresentadas, pode apresentar desempenho prático inferior a decomposição por autovalores. Pensando nestas situações, poderia ser interessante permitir que ao menos a parte quadrática da função objetivo $q_0(x)$ possa ser decomposta através dessa última decomposição, caso a mesma se mostre mais vantajosa, pois a aproximação da função objetivo tem peso considerável na qualidade do limitante inferior fornecido pela relaxação convexa. Ressaltamos que bons limitantes inferiores nos permitem descartar mais precocemente as subpor-

ções do espaço de soluções que não sejam promissoras, economizando assim muitas iterações do algoritmo.

Desse modo, podemos considerar os vetores canônicos e_i, juntamente com os autovetores relacionados a autovalores negativos de Q_0, para compor nosso conjunto de vetores utilizados para construir cada matriz R_j. Assim, sejam $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_d < 0$ os d autovalores negativos de Q_0 e u_i, $i = 1,2,\ldots,d$ seus respectivos autovetores. Utilizamos então o modelo (MIX) (apresentado na Subseção 7.7.1) para a construção de uma decomposição $Q_j = P_j - R_j$ para $j = 0,\ldots,t$. Definimos então:

$$\begin{array}{rcl} v_i & := & \left\{ \begin{array}{ll} u_i, & \text{se } i \leq d, \\ e_{i-d}, & \text{caso contrário,} \end{array} \right. \\ \hat{l}_i & := & \min\{v_i' x \ : \ x \in \mathcal{X}\}, \\ \hat{u}_i & := & \max\{v_i' x \ : \ x \in \mathcal{X}\}, \end{array} \tag{7.23}$$

juntamente com $U_i = u_i u_i'$ e $E_i = e_i e_i'$. Assim, resolvemos, para cada matriz Q_j, o seguinte problema de PSD:

$$\begin{array}{lll} \min\limits_{\lambda \in \mathbb{R}^{d+n}} & \displaystyle\sum_{i=1}^{d+n} \left(\frac{\hat{u}_i^2 + \hat{l}_i^2}{4} - \frac{\hat{u}_i \hat{l}_i}{2} \right) \lambda_i \ , & \\ \text{sujeito a:} & Q_j \ + \ \displaystyle\sum_{i=1}^{d} \lambda_i U_i \ + \ \sum_{i=1}^{n} \lambda_{d+i} E_i \ \succeq 0 \ , & \\ & \lambda \geq 0 \ . & \end{array} \tag{7.24}$$

Após cada resolução acima, tomamos $R_j = \sum_{i=1}^{d} \lambda_i U_i + \sum_{i=1}^{n} \lambda_{d+i} E_i$ e $P_j = Q_j + R_j$. Note que, uma vez que já estamos considerando os vetores $u_i, i = 1,\ldots,d$ para possivelmente serem utilizados na composição de R_0, aproveitamos para permitir que qualquer outro R_j possa ser construído utilizando esses mesmos vetores, caso isto venha a se mostrar oportuno.

Tendo decomposto cada matriz Q_j, construímos então a nossa relaxação convexa para o problema (PQR). Para $j = 0,\ldots,t$ seja $\mu^j(\theta)$ o autovalor de R_j associado ao autovetor θ de R_j. Assim, para $j = 0,\ldots,t$ e $i = 1,\ldots,n$, definimos:

$$\lambda_i^j := \left\{ \begin{array}{ll} \mu^j(v_i) & \text{, se } v_i \text{ é autovetor de } R_j, \\ 0 & \text{, caso contrário} \end{array} \right. \tag{7.25}$$

Por fim, seja $\mathcal{A}^+$ o conjunto de índices i para os quais existe algum $\lambda_i^j > 0$, para $j = 0,\ldots,t$, isto é, $\mathcal{A}^+ = \{i : \exists \lambda_i^j > 0\}$. Aplicando agora o mesmo raciocínio da Seção 7.3, utilizamos a definição de y_i dada pela expressão (7.4) para reformular o problema (PQR) como:

$$
\begin{array}{lll}
(P\bar{Q}R) \quad \min\limits_{x,\,y} & f_0(x) + \frac{1}{2}x'P_0x - \frac{1}{2}\sum\limits_{i\in\mathcal{A}^+}\lambda_i^0 y_i^2 & \\
\text{sujeito a:} & x \in \mathcal{X}, & \\
& h_j(x) + \frac{1}{2}x'P_jx - \frac{1}{2}\sum\limits_{i\in\mathcal{A}^+}\lambda_i^j y_i^2 \leq c_j, \quad j = 1,\ldots,t & \\
& y_i = v_i'x, & i \in \mathcal{A}^+, \\
& l_{y_i} \leq y_i \leq u_{y_i}, & i \in \mathcal{A}^+, \\
& y \in \mathbb{R}^{|\mathcal{A}^+|}, &
\end{array}
\tag{7.26}
$$

onde $l_{y_i} := \hat{l}_i$ e $u_{y_i} := \hat{u}_i$, para $i \in \mathcal{A}^+$. Realizando então a substituição de cada termo côncavo $-y_i^2$ por w_i e introduzindo as desigualdades secantes (7.9), estamos prontos para definir nossa relaxação convexa para o problema (PQR), já omitindo as variáveis y_i:

$$
\begin{array}{lll}
(\underline{PQR}) \quad \min\limits_{x,\,w} & f_0(x) + \frac{1}{2}x'P_0x + \frac{1}{2}\sum\limits_{i\in\mathcal{A}^+}\lambda_i^0 w_i & \\
\text{sujeito a:} & x \in \mathcal{X}, & \\
& h_j(x) + \frac{1}{2}x'P_jx + \frac{1}{2}\sum\limits_{i\in\mathcal{A}^+}\lambda_i^j w_i \leq c_j, \quad j = 1,\ldots,t & \\
& -(u_{y_i} + l_{y_i})v_i'x + l_{y_i}u_{y_i} \leq w_i, & i \in \mathcal{A}^+, \\
& l_{y_i} \leq v_i'x \leq u_{y_i}, & i \in \mathcal{A}^+, \\
& w \in \mathbb{R}^{|\mathcal{A}^+|}, &
\end{array}
\tag{7.27}
$$

Durante a aplicação do BBE, pode ser dada prioridade para a melhoria da aproximação da função objetivo no momento de escolha da variável sobre a qual será feita ramificação. A justificativa para tal decorre do fato de que a aproximação da função objetivo é crucial para a execução de podas por limite. Sejam $(\underline{PQR}^k)$ a relaxação convexa construída na subporção k do espaço de soluções viáveis de $(\underline{PQR})$, e, $(\underline{x}^k, \underline{w}^k)$ a solução ótima dessa relaxação (caso a mesma seja viável). Se a diferença entre a função objetivo de (PQR) e $(\underline{PQR}^k)$ em $(\underline{x}^k, \underline{w}^k)$ for maior que $\underline{\epsilon}$, onde $\underline{\epsilon}$ é um parâmetro positivo próximo a zero, pode-se escolher para ramificação o índice que apresente a maior discrepância entre $-\lambda_i^0(v_i'\underline{x}^k)^2$ e $\lambda_i^0\underline{w}_i^k$, para $i \in \mathcal{A}^+$ (maior discrepância entre o valor do termo côncavo na função objetivo e sua aproximação). Caso contrário, isto é, se a função objetivo de (PQR) está adequadamente aproximada em $(\underline{PQR}^k)$ na solução $(\underline{x}^k, \underline{w}^k)$, podemos escolher, para $i \in \mathcal{A}^+$, o índice que apresente a maior discrepância entre $-\lambda_i^a(v_i'\underline{x}^k)^2$ e $\lambda_i^a\underline{w}_i^k$, onde a é o índice da restrição de (PQR) mais violada em $(\underline{x}^k, \underline{w}^k)$ (maior discrepância entre o valor do termo côncavo na restrição mais violada e sua aproximação). Observe que, se $(\underline{x}^k, \underline{w}^k)$ for viável para (PQR) e não houver diferença entre as funções objetivo de (PQR) e $(\underline{PQR}^k)$ nessa solução, a respectiva subregião será podada por otimalidade

(limite) e a ramificação não será realizada. Para a decisão de como a divisão do espaço será feita sobre o índice escolhido ou sobre a próxima subporção do espaço a ser explorada, podem ser adotados os critérios discutidos na Seção 7.4.

7.8.2 Considerando Variáveis Inteiras

A abordagem aqui apresentada pode ser facilmente estendida para problemas que apresentem funções quadráticas não convexas juntamente com variáveis em domínio discreto (inteiro). Nessa situação, as variáveis inteiras podem ter prioridade sobre as variáveis auxiliares y_i no momento de escolha da variável sobre a qual a ramificação será feita. Para as variáveis inteiras, a ramificação padrão do BB de programação inteira pode ser adotada (veja a Seção 4.3 no Capítulo 4), estando atento ao fato de que, se algum vetor v_i for uma das colunas da matriz identidade e_j com x_j sendo variável inteira no problema abordado, a ramificação padrão do BB de programação inteira pode ser também estendida a y_j a cada ramificação de x_j, conforme discutido na Seção 7.5. Ramificações sobre variáveis inteiras podem eventualmente reduzir os limitantes $l^k_{y_i}$ e $u^k_{y_i}$, mas o esforço para recalcular esses limitantes de forma rigorosa a cada nova subporção do espaço explorada pode prejudicar a eficiência do algoritmo no lugar de trazer ganhos. Todavia, pode valer a pena a aplicação de uma rotina de preprocessamento leve que, a cada iteração de BBE, recalcule esses limitantes para cada y_i apenas com base nas expressões:

$$\begin{aligned} y_i &:= v_i' x, \quad i \in \mathcal{A}^+ \\ l^k_x &\leq x \leq u^k_x, \end{aligned} \tag{7.28}$$

onde l^k_x e u^k_x são os limitantes de x na k-ésima subporção do espaço. Note que, uma vez que agora o problema possui variáveis inteiras, as ramificações também ocorrem sobre x e, assim, a cada subporção do espaço esta variável poderá ter limitantes diferentes. Assim, dados $l^k_{y_i}$ e $u^k_{y_i}$ os limitantes de y_i na k-ésima subporção do espaço, podemos calcular novos limitantes $\hat{l}^k_{y_i}$ e $\hat{u}^k_{y_i}$ de acordo com as expressões:

$$\begin{aligned} \hat{l}^k_{y_i} &= \max\left\{ l^k_{y_i},\ \xi^k_i \right\}, \\ \hat{u}^k_{y_i} &= \min\left\{ u^k_{y_i},\ \bar{\xi}^k_i \right\}, \end{aligned}$$

onde ξ^k_i e $\bar{\xi}^k_i$ são, respectivamente, limitantes inferior e superior para y_i calculados com base apenas em (7.28), isto é:

$$\begin{aligned} \xi^k_i &= \sum_{j\,:\,v_{ij}>0} v_{ij} l^k_{x_j} + \sum_{j\,:\,v_{ij}<0} v_{ij} u^k_{x_j}, \\ \bar{\xi}^k_i &= \sum_{j\,:\,v_{ij}>0} v_{ij} u^k_{x_j} + \sum_{j\,:\,v_{ij}<0} v_{ij} l^k_{x_j}. \end{aligned}$$

Uma possível redução nos limitantes de y_i poderia ser benéfica para melhorar a apro-

ximação secante dos termos côncavos $-y_i^2$, obtendo assim uma relaxação convexa mais forte.

Capítulo 8

Resultados e Contribuições Computacionais

8.1 Resultados Computacionais

Neste seção, apresentamos resultados computacionais para nossa abordagem de resolução de problemas envolvendo funções quadráticas não convexas. Os testes rodaram em um computador com processador core i7 4790 (3,6 GHz), 16 GB de memória RAM sob o sistema operacional Open Suse Linux 13.1 [44]. As implementações foram feitas na linguagem C++ utilizando o compilador `GCC 4.8.1` [71]. Para as resoluções das relaxações convexas no algoritmo de BBE bem como do problema de PSD demandado pela Decomposição Diagonal PSD, foi utilizado pacote `Mosek 7.1.0.33` [48]. As decomposições por autovalores foram calculadas por meio do pacote `LAPACK 3.5.0` [72] com `BLAS 3.5.0` [73]. Consideramos aqui dois cenários distintos para nossos testes. No primeiro cenário, avaliamos diferentes formas de decomposição para problemas de programação quadrática contínua onde apenas a função objetivo é não convexa. No segundo cenário, avaliamos a performance em um subconjunto de instâncias de teste onde introduzimos variáveis inteiras. Posteriormente, comparamos nossos melhores resultados com os conhecidos pacotes `Cplex 12.6.0.0` [47] e `Couenne 0.5.6` [74]. Em todos os contextos, cada diferente execução sobre cada instância de teste foi limitada ao tempo máximo de duas horas. As tolerâncias de convergência absoluta e relativa foram ajustadas em 10^{-4} e 10^{-3}, respectivamente. A estratégia de exploração utilizada no BBE foi a de *menor limitante inferior*, o que significa dizer que, a cada iteração, a subporção em aberto do espaço com o menor limitante inferior era tomada para exploração. Para a escolha do índice para a realização de ramificação, escolhemos o de maior discrepância entre o termo concavo na função objetivo e sua aproximação. O parâmetro α (Equação (7.13)) foi ajustado em 0,8. Todas as aplicações computacionais foram configuradas

para serem executadas por uma única *thread* de processamento, o que significa dizer que as mesmas foram executadas por um único processador a cada instante de tempo na máquina utilizada para os testes.

8.1.1 Avaliando decomposições sobre problemas quadráticos não convexos

Neste contexto, avaliamos o desempenho de nossa abordagem sobre problemas quadráticos não convexos. Os problemas teste não possuem termos não lineares nem restrições quadráticas, apenas uma função objetivo quadrática não convexa sujeita a restrições de caixa e, possivelmente, restrições lineares adicionais. Tomamos aqui os seguintes grupos de instâncias de teste:

- *Relaxed BiqMac (R-BiqMac)*: conjunto de 343 problemas oriundos da biblioteca BiqMac (*Binary quadratic and Max-cut*) [75], onde as restrições de integralidade são relaxadas. Estas instâncias apresentam n variando de 30 até 500 e só possuem restrições de caixa;

- *BoxQP*: conjunto padrão comumente adotado na literatura para comparação de métodos de programação quadrática não convexa. Abrange 99 instâncias de teste quadráticas, com n variando de 20 até 100, com apenas restrições de caixa geradas de forma aleatória em [76];

- *GLOBALLib*: conjunto de instâncias oriundas do conhecido repositório de problemas de otimização global GLOBALLib [77]. Das 413 instâncias disponíveis, selecionamos 83 com função objetivo quadrática não convexa e restrições lineares. O número de variáveis das mesmas varia de 2 até 79;

- *Random*: conjunto de instâncias com função objetivo quadrática não convexa e restrições lineares geradas aleatoriamente para este trabalho. Utilizamos a técnica proposta em [78] para a geração de instâncias de cinco tamanhos diferentes (n = 20,40,60,80,100) em três subgrupos:

  - Problemas cuja função objetivo é composta pela soma de uma função de termos bilineares com uma função quadrática convexa. Cada uma dessas funções é construída sobre um subconjunto disjunto de $\frac{n}{2}$ variáveis;
  - Problemas cuja função objetivo é soma de uma função quadrática côncava com uma função quadrática convexa. Cada uma dessas funções é construída sobre um subconjunto disjunto de $\frac{n}{2}$ variáveis;
  - Problemas cuja função objetivo é a soma de uma função de termos bilineares com uma função quadrática côncava e uma função quadrática

convexa. Cada uma dessas funções é construída sobre um subconjunto disjunto de cerca de $\frac{n}{3}$ variáveis;

Foram geradas quatro instâncias para cada tamanho n em cada grupo, totalizando assim 60 instâncias de teste. O número de restrições lineares em cada problema é $1{,}5n$.

Para cada uma das instâncias acima, testamos as seguintes estratégias de decomposição:

- Decomposição por Autovalores (DA);
- Diagonalmente Dominante (D-DOM);
- Identidade (Ident);
- Diagonal PSD (D-PSD).

Tabela 8.1: Percentual de instâncias resolvidas por cada estratégia de decomposição em cada grupo (%), com tempo máximo de execução de duas horas para cada estratégia em cada instância.

Grupo	**Estratégia de Decomposição**			
(#inst)	**DA**	**D-PSD**	**D-DOM**	**Ident**
biqmac (343)	0,87	16,62	4,37	3,50
boxqp (99)	17,17	68,69	14,14	52,53
globallib (83)	100,00	100,00	97,59	100,00
random (60)	83,33	33,33	16,67	1,67

A Tabela 8.1 apresenta o percentual de instâncias resolvidas por cada estratégia de decomposição em cada grupo no tempo limite estipulado (duas horas). As Figuras 8.1-8.4 trazem gráficos comparativos das estratégias para cada grupo de instâncias. A Figura 8.1 traz a comparação quanto ao tempo computacional tomado para as instâncias totalmente resolvidas no tempo limite estipulado. As Figuras 8.2 e 8.3 comparam, respectivamente, os limitantes inferiores e superiores finais alcançados por cada estratégia. Por fim, a Figura 8.4 compara as estratégias quanto ao limitante superior obtido na primeira iteração do algoritmo de BBE, uma vez que, como as instâncias tomadas possuem apenas restrições convexas, sempre conseguimos uma solução viável para as mesmas a partir da resolução da primeira relaxação convexa no algoritmo de BBE.

Os gráficos nas Figuras 8.1-8.4 apresentam dados normalizados em relação aos melhores resultados obtidos para cada instância. No eixo das abscissas (x), temos valores discriminados para o número de vezes pior que o melhor resultado apresentado dentre todos as estratégias. No eixo das ordenadas, temos o percentual de

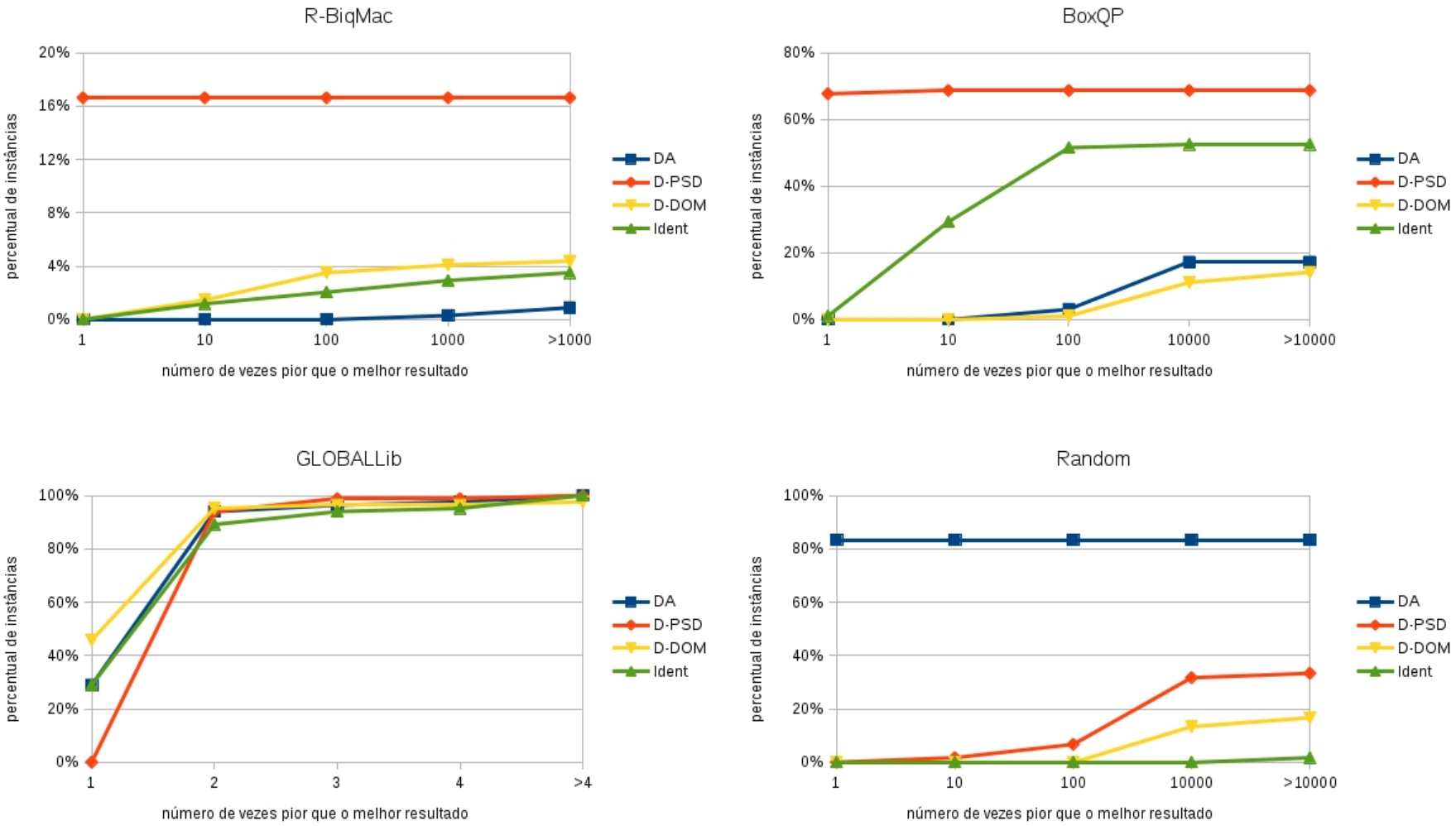


Figura 8.1: Tempo computacional da aplicação BBE segundo estratégias de decomposição para os problemas totalmente resolvidos

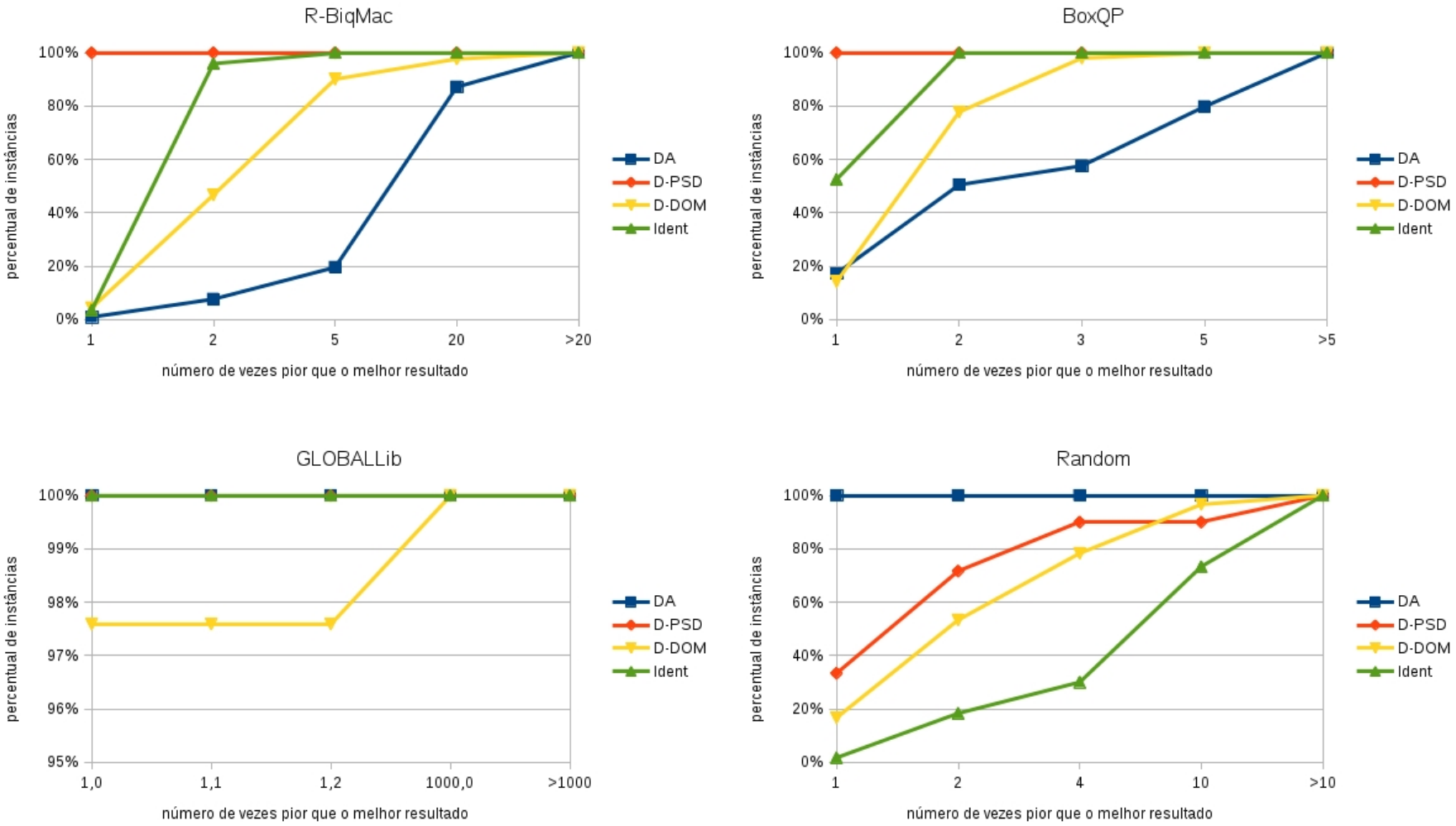


Figura 8.2: Limite inferior obtido com a aplicação BBE segundo estratégias de decomposição

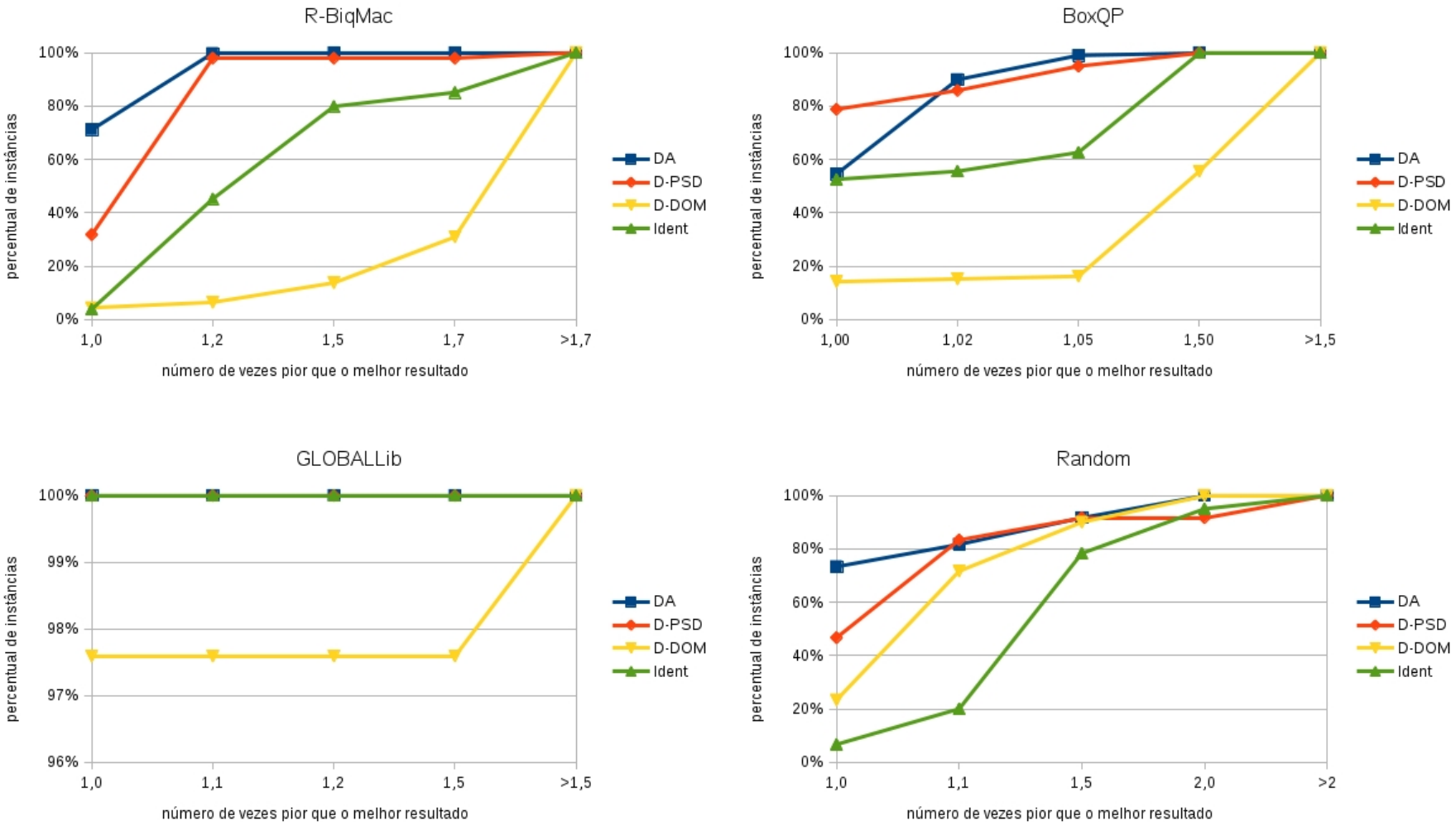


Figura 8.3: Limite superior obtido com a aplicação BBE segundo estratégias de decomposição

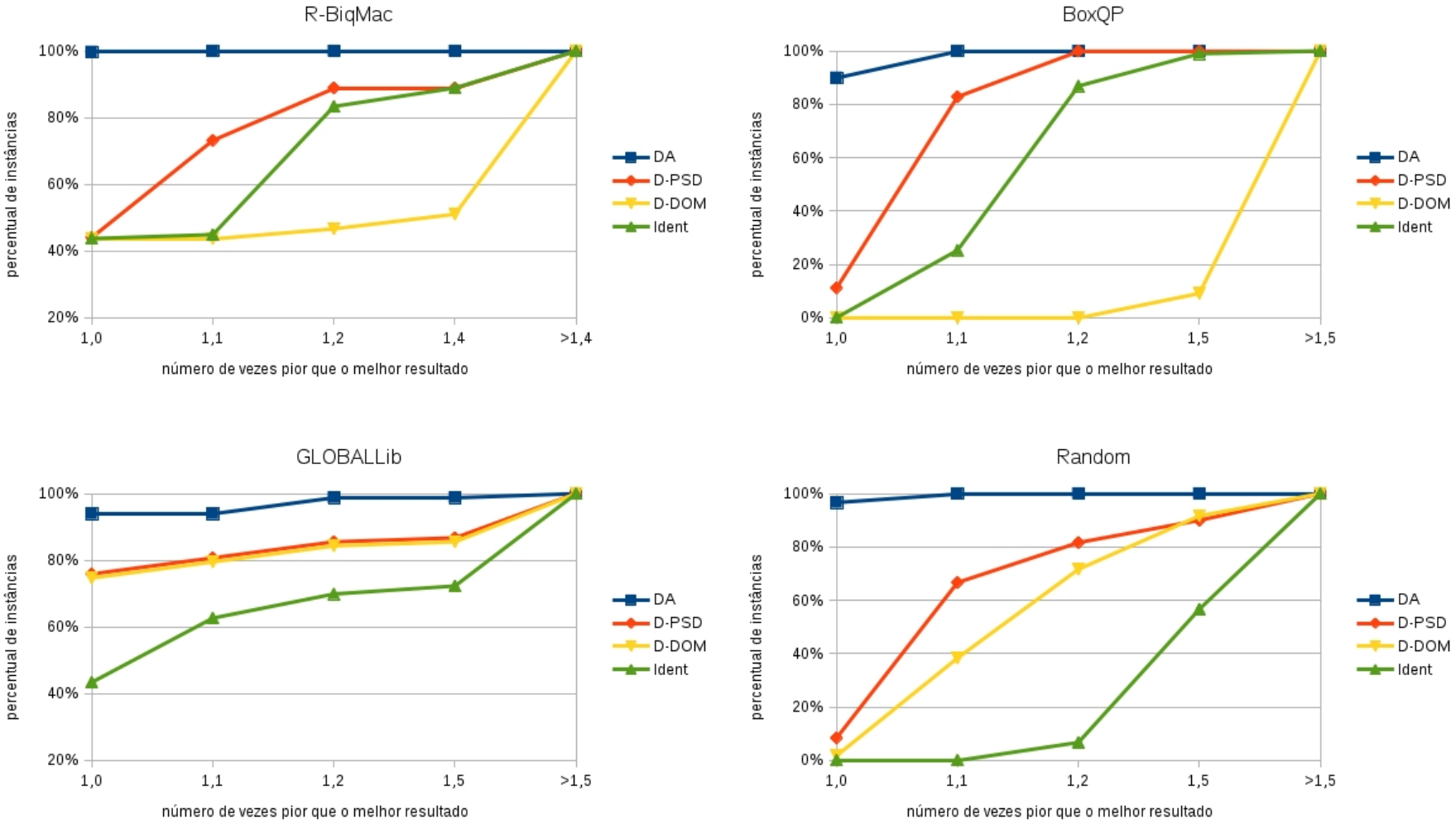


Figura 8.4: Limite superior obtido na primeira iteração com a aplicação BBE segundo estratégias de decomposição

instâncias alcançado por cada abordagem, isto é, o percentual de instâncias onde o resultado obtido por cada estratégia é menor ou igual a x vezes o melhor resultado. Note, por exemplo, que, na Figura 8.1, comparamos as estratégias de decomposição quanto ao tempo computacional. Observe que, para o grupo BoxQP, a curva de Ident passa pelo ponto (1, 0%), indicando que Ident foi capaz de resolver 0% das instâncias desse grupo com menor tempo computacional dentre todas as abordagens (1 vez pior que o melhor resultado). Em seguida, a curva de Ident passa pelo ponto (10, 29%), significando que Ident consegue resolver 29% das instâncias desse grupo com tempo computacional até 10 vezes mais alto que o da melhor estratégia. O próximo ponto da curva é (100, 52%), indicando que Ident consegue resolver 52% das instâncias desse grupo com tempo computacional até 100 vezes mais alto que o da melhor estratégia. Esse percentual alcançado por Ident praticamente se mantém quando se considera tempo computacional acima de 100 vezes maior que o da melhor estratégia.

Analisando a Tabela 8.1 e a Figura 8.1, pode-se notar que, dentre os grupos de instâncias, os piores resultados foram obtidos para as instâncias R-BiqMac, onde foi possível resolver apenas 16,62% das instâncias com D-PSD, que foi a melhor estratégia de decomposição para este grupo. Em parte, esses resultados ruins para esse conjunto de instâncias podem ser explicados pelo número relativamente alto de variáveis presentes nas mesmas (a média e a mediana de n nesse grupo são de 147 e 100, respectivamente).

Podemos destacar que a decomposição D-PSD também apresentou os melhores resultados para o grupo BoxQP, que, assim como R-BiqMac, é o outro grupo onde apenas restrições de caixa estão presentes. Para estes dois grupos de instâncias, podemos observar pelas figuras que D-PSD apresenta o melhor tempo computacional nas instâncias resolvidas bem como o melhor limitante inferior. É interessante observar entretanto, que, DA apresentou o pior desempenho para R-BiqMac com relação ao tempo computacional, número de instâncias resolvidas e limitante inferior. Com relação ao grupo BoxQP, DA superou apenas D-DOM nos quesitos número de instâncias resolvidas e tempo computacional. Todavia, apesar destes resultados ruins em relação aos problemas com restrições de caixa apenas, DA apresentou os melhores resultados no que diz respeito ao limitante superior final encontrado para os problemas R-BiqMac e Random e foi bastante competitivo com D-PSD nesse mesmo quesito para BoxQP e GLOBALLib. Analisando o limitante superior obtido na primeira iteração, é notável que DA superou D-PSD nos grupos R-BiqMac e BoxQP apresentando ainda os melhores resultados nesse quesito para todos os grupos de instâncias. Esse comportamento nos indica que a estratégia DA pode ser mais efetiva para conduzir as relaxações a proverem boas soluções viáveis, mesmo nos casos onde os limitantes inferiores obtidos, assim como o desempenho de modo

geral, são ruins.

É possível perceber ainda que DA se saiu bem nos grupos GLOBALLib e Random, que são compostos de instâncias com restrições lineares. Nesse último grupo, DA apresentou performance notavelmente superior às decomposições diagonais. Conforme esperado para estas últimas decomposições, em geral, D-PSD domina o desempenho de D-DOM e Ident. No geral, os problemas de PSD necessários à D-PSD foram resolvidos sem maiores dificuldades (mesmo para as instâncias maiores, com $n = 500$, este problema sempre pôde ser resolvido em não mais que 15 segundos em nosso ambiente de testes). Entretanto, para problemas de dimensões consideravelmente maiores, a resolução desses problemas pode se tornar excessivamente dispendiosa para ser realizada em tempo hábil. Todavia, em muitos casos, Ident se apresenta como uma boa decomposição alternativa a D-PSD quando um processo de decomposição rápido é requisitado.

As conclusões acima nos dão uma indicação de que as decomposições diagonais, em especial D-PSD, podem ser mais efetivas em problemas com apenas restrições de caixa, ao passo que DA pode ser mais efetiva em problemas com restrições mais gerais. Uma possível explicação para esse comportamento vem do fato de que para a reformulação de (PQ) como um problema separável, introduzimos as variáveis auxiliares $y_i := v_i' x,\ i \in \mathcal{A}^+$. Para as decomposições diagonais, $v_i = e_i$ e as varáveis y_i podem herdar os limitantes de x_i, isto, é $l_{y_i} = l_{x_i}$ e $u_{y_i} = u_{x_i}$. Por outro, para DA, em geral, cada v_i pode possuir mais de um elemento não nulo. Nesse, caso, como não existem restrições além das de caixa, os limitantes l_{y_i} e u_{y_i} precisam calculados a partir dos limitantes l_{x_j} e u_{x_j} segundo a expressão:

$$\begin{aligned} l_{y_i} &= \sum_{j\,:\,v_{ij}>0} v_{ij} l_{x_j} + \sum_{j\,:\,v_{ij}<0} v_{ij} u_{x_j}\ , \\ u_{y_i} &= \sum_{j\,:\,v_{ij}>0} v_{ij} u_{x_j} + \sum_{j\,:\,v_{ij}<0} v_{ij} l_{x_j}\ , \end{aligned} \tag{8.1}$$

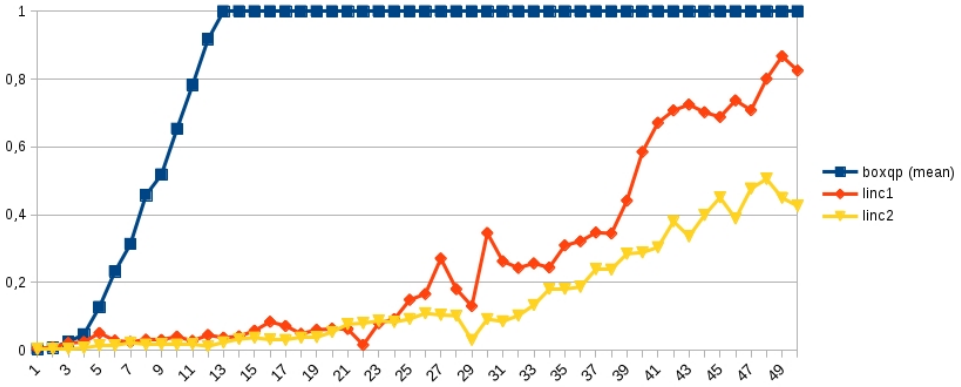
para $i \in \mathcal{A}^+$. Desta maneira, os limitantes l_{y_i} e u_{y_i} tendem a definir, na decomposição DA, caixas de comprimentos maiores do que as definidas nas decomposições diagonais. Assim, as aproximações secantes construídas na decomposição DA terão de cobrir um intervalo maior que as aproximações secantes das decomposições diagonais. Esse intervalo maior tende a gerar uma aproximação de qualidade inferior. Ademais, mais divisões do espaço tendem a serem necessárias para que as desigualdades secantes consigam provocar uma aproximação adequada do problema sendo tratado. Estes fatos podem estar contribuindo para um pior desempenho por parte da decomposição DA em relação às decomposições diagonais nos problemas com restrições de caixa, embora, a decomposição DA tenha a vantagem de poder gerar uma matriz R_0 cheia, e, assim, ser capaz de extrair um maior nível de convexidade de $q_0(x)$. Em problemas com restrições mais gerais, por sua vez, os limitantes l_{y_i} e

u_{y_i} da decomposição DA tendem a definir uma caixa de comprimento menor que os limitantes calculados com a expressão (8.1), o que tende a melhorar as aproximações secantes de $-\frac{1}{2}y^2$, contribuindo assim para um melhor desempenho dessa estratégia de decomposição.

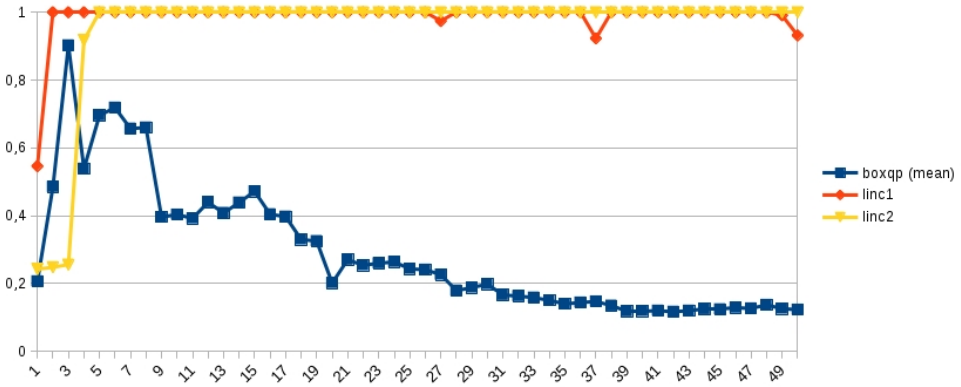
Avaliando os efeitos da variação do número de autovalores negativos de Q_0

Uma série de experimentos foram realizados com o intuito de avaliar como as estratégias de decomposição se comportam ao se variar o grau de não convexidade quadrática na função objetivo. Foram geradas instâncias aleatórias variando-se o número de autovalores negativos da matriz Q_0. Primeiramente, geramos quatro instâncias básicas com n = 50, duas delas com apenas restrições de caixa (boxqp1 e boxqp2) e duas delas com restrições lineares (linc1 e linc2). A seguir, alteramos os sinais dos autovalores das matrizes Q_0 em cada instância (preservando os autovetores), gerando assim quatro grupos de 50 instâncias, cada, com o número de autovalores negativos de Q_0 em cada grupo variando de 1 a 50. Aplicamos então nossa abordagem de BBE sobre cada grupo com as decomposições DA (cheia) e D-PSD (nossa melhor decomposição diagonal). A Figura 8.5 traz gráfcos que avaliam o desempenho dessas duas estratégias de decomposição, para os diferentes números de autovalores negativos em cada grupo. O eixo das abscissas indica o número de autovalores negativos de Q_0, ao passo que o eixo das ordenadas indica a fração do tempo máximo de processamento (2 horas) para resolver cada instância, calculado em escala logarítmica. Uma vez que os resultados obtidos para os grupos boxqp1 e boxqp2 foram muito similares, apenas uma curva representando a média do resultados desses dois grupos é exibida em cada gráfico.

Podemos perceber, por meio da Figura 8.5, que o desempenho da estratégia DA apresenta forte relação com o número de autovalores negativos de Q_0. É notável que, quanto maior este número, maior o esforço computacional despendido por essa estratégia. Podemos ressaltar que, nessa decomposição, o número de autovalores negativos de Q_0 determina o número de autovalores não nulos (positivos) de R_0, e, por consequência, o número de termos côncavos $-\frac{1}{2}y_i^2$ aproximados por meio de secantes. Em oposição, o gráfico de D-PSD não sugere uma relação tão evidente entre o seu desempenho e o número de autovalores negativos. O gráfico curiosamente indica que, para os problemas com restrição de caixa apenas (boxqp1 e boxqp2), números elevados de autovalores negativos ajudam na performance dessa estratégia de decomposição. Vale destacar que, na estratégia D-PSD, o número de termos côncavos $-\frac{1}{2}y_i^2$ não possui uma relação tão direta com o número de autovalores negativos de Q_0 como em DA. Em geral, decomposições diagonais tendem a gerar R_0 com um número elevado de autovalores não nulos (positivos), mesmo em casos



a Decomposição por Autovalores (DA)



b Decomposição Diagonal PSD (D-PSD)

Figura 8.5: Efeito do número de autovalores negativos de Q_0 nas decomposições DA e D-PSD.

onde Q_0 apresenta poucos autovalores negativos, o que poderia ajudar a explicar o comportamento observado.

Para problemas com baixo número de autovalores negativos em Q_0 (8 ou menos), DA é sempre a melhor estratégia para todas instâncias nos quatro grupos. Quando o número de autovalores negativos aumenta, a melhor estratégia depende da categoria das instâncias. Para instâncias com restrições lineares (linc1 e linc2), DA supera, de forma significativa, o desempenho de D-PSD. Para problemas com restrições de caixa, D-PSD é a melhor estratégia exceto nos casos onde o número de autovalores negativos de Q_0 é muito baixo.

8.1.2 Avaliando desempenho em problemas com variáveis inteiras

Descrevemos a seguir experimentos realizados com o objetivo de avaliar o desempenho de nossa abordagem em problemas com variáveis inteiras. Geramos um novo grupo de 99 instâncias de problemas quadráticos com restrições de caixa a partir do grupo BoxQP, denominado MIBoxQP. Reescalamos as instâncias desse grupo de

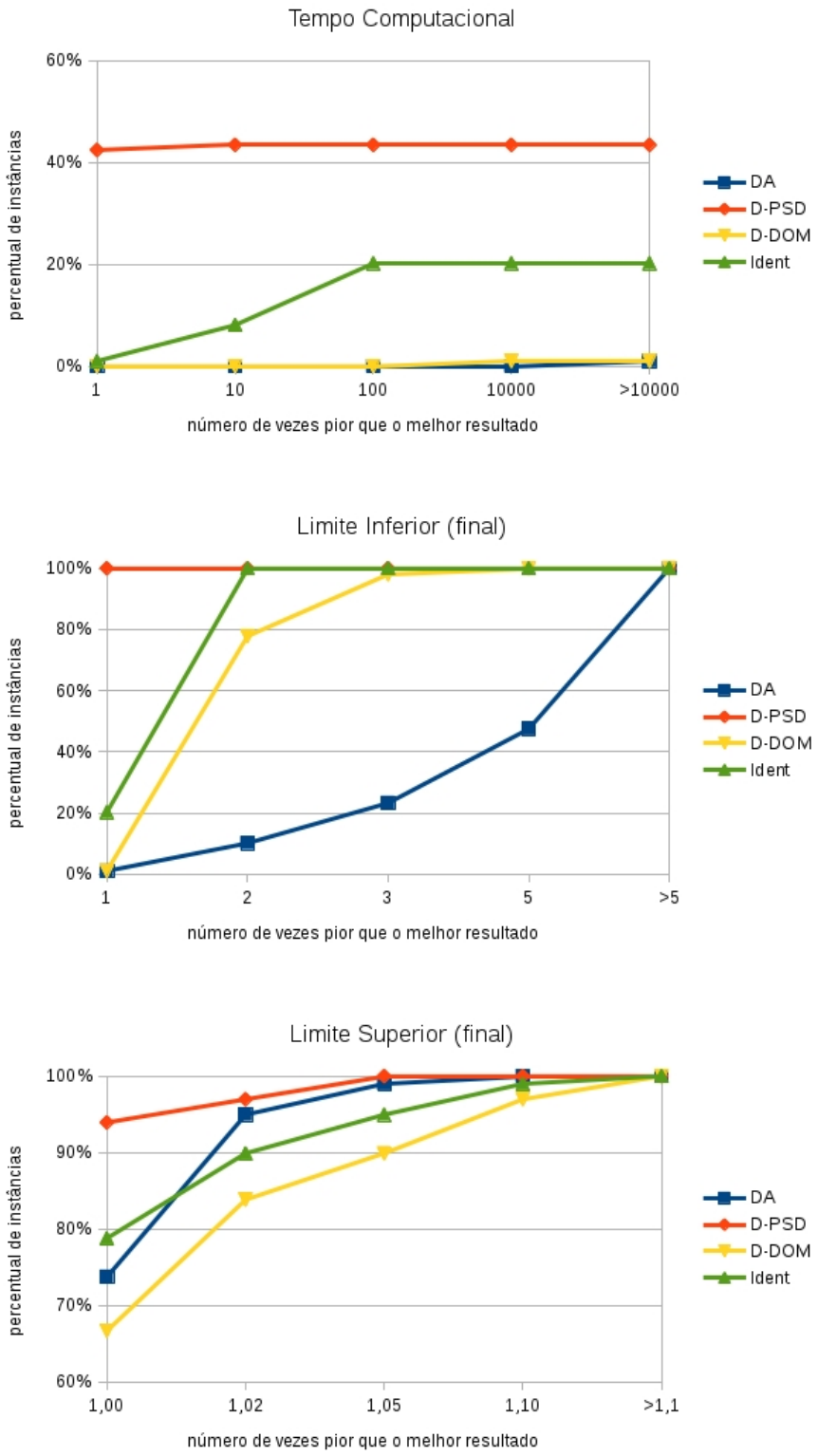


Figura 8.6: Resultados para as diferentes estratégias de decomposição sobre MI-BoxQP.

modo que as variáveis estejam contidas na caixa $[0,\ 10]^n$ (originalmente as variáveis estão na caixa $[0,\ 1]^n$) e impomos integralidade nas primeiras $\frac{n}{2}$ variáveis, mantendo as últimas $\frac{n}{2}$ contínuas.

A Figura 8.6 exibe gráficos comparando as diferentes estratégias de decomposição com respeito ao tempo computacional, limitante inferior final e limitante superior final. Uma vez que essas instâncias possuem variáveis inteiras, a solução obtida na primeira aplicação de BBE não é necessariamente viável. Por esta razão, não faria sentido um gráfico sobre o limitante superior obtido na primeira iteração. O percentual de instâncias resolvidas por cada estratégia foi de 1,01% para AD e D-DOM, 20,20% para Ident e 43,42% para D-PSD. Observamos que estes percentuais são menores em relação aos obtidos para as instâncias BoxQP contínuas. Todavia, de um modo, geral, o comportamento das estratégias de decomposição nesse novo grupo é condizente com o apresentado para BoxQP. D-PSD apresentou os melhores resultados para os quesitos avaliados, com DA sendo bastante competitivo a este no quesito limitante superior final. Estes resultados reforçam a consistência da abordagem geral aqui proposta quando estendida a problemas com variáveis inteiras.

8.1.3 Comparação com pacotes computacionais

Nesta subseção, apresentamos resultados comparativos entre os melhores resultados obtidos em nossa proposta com resultados oriundos dos pacotes computacionais `Cplex 12.6.0.0` e `Couenne 0.5.6`. Denominamos aqui nossa melhor abordagem como `iquad`, que é o nome escolhido para o nosso pacote computacional que implementa a abordagem aqui apresentada. `Cplex` é um respeitado *solver* nas áreas de programação linear e quadrática inteiras mista, e, recentemente, foi estendido para a resolução de problemas quadráticos não convexos também. `Couenne`, por sua vez, é um pacote conhecido para problemas mais gerais de otimização global, onde não convexidades também podem estar presente em funções não lineares. Para esta comparação, selecionamos as 10% instâncias mais difíceis, segundo nossos resultados anteriores, em cada um dos cinco grupos utilizados até aqui (R-BiqMac, BoxQP, GLOBALLib, Random e MIBoxQP). Detalhes sobre essas instâncias podem ser conferidos no Apêndice C. Para a seleção desses subgrupos, escolhemos, em cada grupo, as instâncias onde nossos melhores resultados apresentam os maiores *gaps* de dualidade normalizado após o tempo limite de execução.

Com base nos resultados obtidos nos experimentos anteriores, `iquad` foi ajustado para utilizar decomposição DA em instâncias com restrições lineares e D-PSD em instâncias com apenas restrições de caixa. De um modo geral, as abordagens não foram capazes de solucionar as instâncias até a otimalidade no tempo limite de duas horas. As exceções são as instâncias oriundas de GLOBALLib e duas instâncias

de R-BiqMac que foram resolvidas por `Cplex` e `Couenne` e não foram resolvidas por `iquad`. Em contrapartida, `iquad` resolveu todas as instâncias GLOBALLib, ao passo que tanto `Cplex` quanto `Couenne` não puderam resolver duas dessas instâncias.

Tabela 8.2: Comparação entre nossa abordagem (iquad), Couenne e Cplex - *Gap* de dualidade normalizado após duas horas de execução (%).

Grupo (#Inst)	`iquad`	`Couenne`	`Cplex`
R-BiqMac (35)	23,10	90,51	87,45
BoxQP (10)	5,87	100,00	95,94
GLOBALLib (9)	0,00	22,22	4,29
Random (6)	0,07	100,00	6,86
MIBoxQP (10)	5,87	100,00	95,94

A Tabela 8.2 traz a média do *gap* de dualidade normalizado de cada abordagem para cada grupo de instâncias. Este *gap* de dualidade normalizado é definido como a percentagem do pior *gap* de dualidade entre os três *solvers* (`iquad`, `Couenne` e `Cplex`). Por exemplo, para o *solver* i, o *gap* de dualidade normalizado é dado por $\frac{ub_i - lb_i}{(ub-lb)_{max}} \times 100$, onde ub_i e lb_i são, respectivamente, os limitantes inferior e superior calculados pelo *solver* i, e $(ub - lb)_{max}$ é o pior *gap* de dualidade obtido dentre todos os *solvers*. O número total de instâncias em cada subgrupo também é mostrado nessa tabela.

Podemos observar na Tabela 8.2 que os *gaps* de `iquad` e `Cplex` no subgrupo Random são pequenos, embora nenhuma instância desse subgrupo tenha sido resolvida. Isto ocorre devido ao fato de `Couenne` ter retornado -10^{20} como limitante inferior em 5 das 6 instâncias desse subgrupo. Pode-se notar ainda que `iquad` apresentou os melhores resultados para todos os subgrupos dentre os três pacotes considerados, reforçando assim a relevância e a efetividade do trabalho desenvolvido aqui, que foi capaz de superar a pacotes considerados como estado da arte nas instâncias de teste adotadas.

8.2 Contribuição Computacional

Para esta parte do trabalho, desenvolvemos um pacote computacional que implementa a abordagem discutida aqui, com as variadas formas de decomposição DC tomada em nossos testes computacionais. Denominados este pacote como `iquad` (*indefinite quadratics*). O núcleo de `iquad` já conta com cerca de 7.500 linhas de código na linguagem C++ (já excluindo linhas vazias e de comentário). Além disso, o desenvolvimento de `iquad` tem sido facilitado pela incorporação dos componentes reutilizáveis já desenvolvidos para o *solver* `Muriqui`, descritos na Seção 5.2. O sucesso na reutilização desses componentes é um indicativo de bom nível de versati-

lidade e qualidade presentes no *software* construído em nosso trabalho de pesquisa como um todo.

Desde o início do projeto, `iquad` teve como um de seus objetivos a sua aplicação na resolução dos problemas de minimização de *gap* do algoritmo AMGI1 em `Muriqui`. Entretanto, nossa abordagem não se mostrou eficiente em lidar com a função objetivo côncava com termos lineares desses problemas. Em especial, a inexistência de convexidade a ser extraída de Q_0 bem como as aproximações secantes dos termos côncavos, que terminam por contrapor os termos lineares originais da função objetivo, são os principais responsáveis pelo insucesso de `iquad` nessa aplicação. Todavia, ainda temos observado, como perspectiva futura, uma integração cada vez maior entre `Muriqui` e `iquad`. Uma possível linha de atuação é a investigação sobre formas eficientes de resolver os problemas de minimização de *gap* de `Muriqui` tomando como base a pesquisa desenvolvida nesta parte do trabalho. Em contrapartida, também apontamos como trabalho futuro a incorporação de estratégias mais elaboradas para lidar com variáveis inteiras em nossos problemas com funções quadráticas não convexas abordados por `iquad`. Nesse contexto, o aprendizado já obtido com o desenvolvimento de `Muriqui` pode se mostrar oportuno para melhorar a eficiência do BBE implementado em `iquad`. Observamos que uma quantidade maior de sub-rotinas computacionais pode vir a ser compartilhada entre ambos os pacotes. Por fim, ainda apontamos como trabalhos futuros, no âmbito de `iquad`, pesquisas relacionadas à aplicação de nossa abordagem à problemas com restrições quadráticas não convexas, conforme descrito na Subseção 7.8.1.

Referências Bibliográficas

[1] MELO, W. *Agoritmos para Programação Não Linear Inteira Mista*. dissertação de mestrado, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2012.

[2] TRESPALACIOS, F., GROSSMANN, I. E. "Review of Mixed-Integer Nonlinear and Generalized Disjunctive Programming Methods", *Chemie Ingenieur Technik*, v. 86, n. 7, pp. 991–1012, 2014. ISSN: 1522-2640. doi: 10.1002/cite.201400037.

[3] D'AMBROSIO, C., LODI, A. "Mixed integer nonlinear programming tools: a practical overview", *4OR*, v. 9, n. 4, pp. 329–349, 2011. ISSN: 1619-4500. doi: 10.1007/s10288-011-0181-9.

[4] HEMMECKE, R., KÖPPE, M., LEE, J., et al. "Nonlinear Integer Programming". In: Jünger, M., Liebling, T. M., Naddef, D., et al. (Eds.), *50 Years of Integer Programming 1958-2008*, Springer Berlin Heidelberg, pp. 561–618, 2010. ISBN: 978-3-540-68279-0. 10.1007/978-3-540-68279-0_15.

[5] BONAMI, P., KILINÇ, M., LINDEROTH, J. *Algorithms and Software for Convex Mixed Integer Nonlinear Programs*. Relatório Técnico 1664, Computer Sciences Department, University of Wisconsin-Madison, 2009.

[6] BELOTTI, P. "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`.

[7] MOURET, S., GROSSMANN, I. "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`.

[8] LIU, P., PISTIKOPOULOS, E. N., LI, Z. "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.

[9] YOU, F., GROSSMANN, I. E. "Mixed-Integer Nonlinear Programming Models and Algorithms for Large-Scale Supply Chain Design with Stochastic Inventory Management", *Industrial & Engineering Chemistry Research*, v. 47, n. 20, pp. 7802–7817, 2008. doi: 10.1021/ie800257x.

[10] GENTILINI, I. "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.

[11] DURAN, M., GROSSMANN, I. "An outer-approximation algorithm for a class of mixed-integer nonlinear programs", *Mathematical Programming*, v. 36, pp. 307–339, 1986. ISSN: 0025-5610. 10.1007/BF02592064.

[12] CHRISTODOULOU, M., COSTOULAKIS, C. "Nonlinear mixed integer programming for aircraft collision avoidance in free flight". In: *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, v. 1, pp. 327 – 330 Vol.1, may 2004. doi: 10.1109/MELCON.2004.1346858.

[13] GUILLEN, G., POZO, C. "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.

[14] BRAGALLI, C., D'AMBROSIO, C., LEE, J., et al. "On the optimal design of water distribution networks: a practical MINLP approach", *Optimization and Engineering*, v. 13, pp. 219–246, 2012. ISSN: 1389-4420. 10.1007/s11081-011-9141-7.

[15] GOPALAKRISHNAN, A., BIEGLER, L. "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.

[16] LEYFFER, S., LINDEROTH, J., LUEDTKE, J., et al. "Applications and algorithms for mixed integer nonlinear programming", *Journal of Physics: Conference Series*, v. 180, n. 1, pp. 012014, 2009.

[17] GEOFFRION, A. M. "Generalized Benders decomposition", *Journal of Optimization Theory and Applications*, v. 10, pp. 237–260, 1972. ISSN: 0022-3239. 10.1007/BF00934810.

[18] FLETCHER, R., LEYFFER, S. "Solving mixed integer nonlinear programs by outer approximation", *Mathematical Programming*, v. 66, pp. 327–349, 1994. ISSN: 0025-5610. 10.1007/BF01581153.

[19] QUESADA, I., GROSSMANN, I. "An LP/NLP based branch and bound algorithm for convex MINLP optimization problems", *Computers & Chemical Engineering*, v. 16, n. 10-11, pp. 937 – 947, 1992. ISSN: 0098-1354. doi: 10.1016/0098-1354(92)80028-8. An International Journal of Computer Applications in Chemical Engineering.

[20] WESTERLUND, T., PETTERSSON, F. "An extended cutting plane method for solving convex MINLP problems", *Computers & Chemical Engineering*, v. 19, Supplement 1, n. 0, pp. 131 – 136, 1995. ISSN: 0098-1354. doi: 10.1016/0098-1354(95)87027-X. European Symposium on Computer Aided Process Engineering.

[21] GUPTA, O. K., RAVINDRAN, A. "Branch and Bound Experiments in Convex Nonlinear Integer Programming", *Management Science*, v. 31, n. 12, pp. 1533–1546, 1985. doi: 10.1287/mnsc.31.12.1533.

[22] BORCHERS, B., MITCHELL, J. E. "An improved branch and bound algorithm for mixed integer nonlinear programs", *Comput. Oper. Res.*, v. 21, pp. 359–367, April 1994. ISSN: 0305-0548. doi: 10.1016/0305-0548(94)90024-8.

[23] LEYFFER, S. "Integrating SQP and Branch-and-Bound for Mixed Integer Nonlinear Programming", *Comput. Optim. Appl.*, v. 18, pp. 295–309, March 2001. ISSN: 0926-6003. doi: 10.1023/A:1011241421041.

[24] STILL, C., WESTERLUND, T. "Solving Convex MINLP Optimization Problems Using a Sequential Cutting Plane Algorithm", *Computational Optimization and Applications*, v. 34, pp. 63–83, 2006. ISSN: 0926-6003. 10.1007/s10589-005-3076-x.

[25] STUBBS, R. A., MEHROTRA, S. "A branch-and-cut method for 0-1 mixed convex programming", *Mathematical Programming*, v. 86, pp. 515–532, 1999. ISSN: 0025-5610. 10.1007/s101070050103.

[26] BONAMI, P., LEE, J., LEYFFER, S., et al. "On Branching Rules for Convex Mixed-integer Nonlinear Optimization", *J. Exp. Algorithmics*, v. 18, pp. 2.6:2.1–2.6:2.31, nov. 2013. ISSN: 1084-6654. doi: 10.1145/2532568. Disponível em: <http://doi.acm.org/10.1145/2532568>.

[27] EXLER, O., SCHITTKOWSKI, K. "A trust region SQP algorithm for mixed-integer nonlinear programming", *Optimization Letters*, v. 1, pp. 269–280, 2007. ISSN: 1862-4472. doi: 10.1007/s11590-006-0026-1.

[28] EXLER, O., LEHMANN, T., SCHITTKOWSKI, K. "A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization", *Mathematical Programming Computation*, v. 4, pp. 383–412, 2012. ISSN: 1867-2949. doi: 10.1007/s12532-012-0045-0.

[29] LI, D., WANG, J., SUN, X. "Computing exact solution to nonlinear integer programming: Convergent Lagrangian and objective level cut method", *Journal of Global Optimization*, v. 39, pp. 127–154, 2007. ISSN: 0925-5001. doi: 10.1007/s10898-006-9128-7.

[30] MURRAY, W., NG, K.-M. "An algorithm for nonlinear optimization problems with binary variables", *Computational Optimization and Applications*, v. 47, pp. 257–288, 2010. ISSN: 0926-6003. 10.1007/s10589-008-9218-1.

[31] GROSSMANN, I. E., RUIZ, J. P. "Generalized Disjunctive Programming: A Framework for Formulation and Alternative Algorithms for MINLP Optimization". In: Lee, J., Leyffer, S. (Eds.), *Mixed Integer Nonlinear Programming*, v. 154, *The IMA Volumes in Mathematics and its Applications*, Springer New York, pp. 93–115, 2012. ISBN: 978-1-4614-1927-3. 10.1007/978-1-4614-1927-3_4.

[32] LEE, S., GROSSMANN, I. E. "New algorithms for nonlinear generalized disjunctive programming", *Computers & Chemical Engineering*, v. 24, n. 9-10, pp. 2125 – 2141, 2000. ISSN: 0098-1354. doi: 10.1016/S0098-1354(00)00581-0.

[33] GROSSMANN, I. E., LEE, S. "Generalized Convex Disjunctive Programming: Nonlinear Convex Hull Relaxation", *Computational Optimiza-*

tion and Applications, v. 26, pp. 83–100, 2003. ISSN: 0926-6003. 10.1023/A:1025154322278.

[34] RAMAN, R., GROSSMANN, I. "Modelling and computational techniques for logic based integer programming", *Computers & Chemical Engineering*, v. 18, n. 7, pp. 563 – 578, 1994. ISSN: 0098-1354. doi: 10.1016/0098-1354(93)E0010-7. An International Journal of Computer Applications in Chemical Engineering.

[35] BONAMI, P., BIEGLER, L. T., CONN, A. R., et al. "An algorithmic framework for convex mixed integer nonlinear programs", *Discrete Optimization*, v. 5, n. 2, pp. 186–204, maio 2008. doi: 10.1016/j.disopt.2006.10.011.

[36] MELO, W., FAMPA, M., RAUPP, F. "Integrating nonlinear branch-and-bound and outer approximation for convex Mixed Integer Nonlinear Programming", *Journal of Global Optimization*, v. 60, n. 2, pp. 373–389, 2014. ISSN: 0925-5001. doi: 10.1007/s10898-014-0217-8.

[37] BONAMI, P., GONÇALVES, J. A. "Heuristics for convex mixed integer nonlinear programs", *Computational Optimization and Applications*, pp. 1–19, 2008. ISSN: 0926-6003. 10.1007/s10589-010-9350-6.

[38] BONAMI, P., CORNUÉJOLS, G., LODI, A., et al. "A Feasibility Pump for mixed integer nonlinear programs", *Mathematical Programming*, v. 119, pp. 331–352, 2009. ISSN: 0025-5610. 10.1007/s10107-008-0212-2.

[39] BERTHOLD, T., GLEIXNER, A. M. *Undercover - a primal heuristic for MINLP based on sub-MIPs generated by set covering.* Relatório Técnico ZIB-REPORT 09-04, 2009.

[40] LIBERTI, L., MLADENOVIĆ, N., NANNICINI, G. "A recipe for finding good solutions to MINLPs", *Mathematical Programming Computation*, v. 3, pp. 349–390, 2011. ISSN: 1867-2949. 10.1007/s12532-011-0031-y.

[41] NANNICINI, G., BELOTTI, P. "Rounding-based heuristics for nonconvex MINLPs", *Mathematical Programming Computation*, v. 4, pp. 1–31, 2012. ISSN: 1867-2949. 10.1007/s12532-011-0032-x.

[42] RAGHAVACHARI, M. "On Connections Between Zero-One Integer Programming and Concave Programming Under Linear Constraints", *Operations Research*, v. 17, n. 4, pp. 680–684, July/August 1969. doi: 10.1287/opre.17.4.680.

[43] CMU-IBM. "Open source MINLP Project". 2012. Disponível em: <http://egon.cheme.cmu.edu/ibm/page.htm>.

[44] "OPENSUSE - Linux OS". Software. Disponível em: <https://www.opensuse.org/>.

[45] "ICPC: Intel C++ Compiler". Software. Disponível em: <https://software.intel.com/en-us/c-compilers>.

[46] "GAMS, General Algebraic Modeling System". Software. Disponível em: <https://www.gams.com/>.

[47] "ILOG CPLEX". Software. Disponível em: <http://www.ilog.com/products/cplex/>.

[48] "The MOSEK optimization software". Software. Disponível em: <http://www.mosek.com/>.

[49] WÄCHTER, A., BIEGLER, L. T. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical Programming*, v. 106, pp. 25–57, 2006. ISSN: 0025-5610. 10.1007/s10107-004-0559-y.

[50] "HSL. A collection of Fortran codes for large scale scientific computation." Software, 2007. Disponível em: <http://www.hsl.rl.ac.uk/>.

[51] "AMPL, A Modeling Language for Mathematical Programming". Software. Disponível em: <http://www.ampl.com/>.

[52] MELO, W. A., FAMPA, M. H., RAUPP, F. M. "A stochastic local search algorithm for constrained continuous global optimization", *International Transactions in Operational Research*, v. 19, n. 6, pp. 825–846, 2012. ISSN: 1475-3995. doi: 10.1111/j.1475-3995.2012.00854.x. Disponível em: <http://dx.doi.org/10.1111/j.1475-3995.2012.00854.x>.

[53] MELO, W., FAMPA, M., RAUPP, F. "Evolução Diferencial Aperfeiçoada para otimização contínua restrita". In: *XLII Simpósio Brasileiro de Pesquisa Operacional*, Bento Gonçalves, RS, Brasil, 2010.

[54] MELO, W., FAMPA, M., RAUPP, F. "Cutting Box Strategy: An Algorithmic Framework for Improving Metaheuristics for Continuous Global Optimization". In: Michalski, A. (Ed.), *Global Optimization: Theory, Developments and Applications*, Mathematics Research Developments Computational Mathematics and Analysis Series, Nova Science Publishers, pp. 155–176, 2013. ISBN: 978-1-62417-796-5.

[55] FAMPA, M., LEE, J., MELO, W. "A specialized branch-and-bound algorithm for the Euclidean Steiner tree problem in n-space", *Computational Optimization and Applications*, pp. 1–25, 2016. ISSN: 1573-2894. doi: 10.1007/s10589-016-9835-z. Disponível em: <http://dx.doi.org/10.1007/s10589-016-9835-z>.

[56] Pardalos, P., Romeijn, H. E. (Eds.). *Handbook of Global Optimization*, v. 2. Springer, 2002. ISBN: 978-1-4757-5362-2.

[57] Horst, R., Pardalos, P. (Eds.). *Handbook of Global Optimization*, v. 1. Springer, 1995. ISBN: 978-1-4615-2025-2.

[58] FLOUDAS, C. A., GOUNARIS, C. E. "A Review of Recent Advances in Global Optimization", *J. of Global Optimization*, v. 45, n. 1, pp. 3–38, 2009. ISSN: 0925-5001. doi: 10.1007/s10898-008-9332-8. Disponível em: <http://dx.doi.org/10.1007/s10898-008-9332-8>.

[59] RENDL, F., RINALDI, G., WIEGELE, A. "Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations", *Mathematical Programming*, v. 121, n. 2, pp. 307–335, 2008. ISSN: 1436-4646. doi: 10.1007/s10107-008-0235-8. Disponível em: <http://dx.doi.org/10.1007/s10107-008-0235-8>.

[60] BURKARD, R., ÇELA, E., PARDALOS, P., et al. "The Quadratic Assignment Problem". In: Du, D.-Z., Pardalos, P. (Eds.), *Handbook of Combinatorial Optimization*, Springer US, pp. 1713–1809, 1999. ISBN: 978-1-4613-7987-4. doi: 10.1007/978-1-4613-0303-9_27.

[61] LEE, J. "In situ column generation for a cutting-stock problem", *Computers & Operations Research*, v. 34, n. 8, pp. 2345 – 2358, 2007. ISSN: 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2005.09.007. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054805002972>.

[62] SAXENA, A., BONAMI, P., LEE, J. "Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations", *Mathematical Programming*, v. 130, n. 2, pp. 359–413, 2010. ISSN: 1436-4646. doi: 10.1007/s10107-010-0340-3. Disponível em: <http://dx.doi.org/10.1007/s10107-010-0340-3>.

[63] SAXENA, A., BONAMI, P., LEE, J. "Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations", *Mathematical Programming*, v. 124, n. 1, pp. 383–411, 2010. ISSN:

1436-4646. doi: 10.1007/s10107-010-0371-9. Disponível em: <http://dx.doi.org/10.1007/s10107-010-0371-9>.

[64] AN, L. T. H., TAO, P. D. "The DC (Difference of Convex Functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems", *Annals of Operations Research*, v. 133, n. 1, pp. 23–46, 2005. ISSN: 1572-9338. doi: 10.1007/s10479-004-5022-1. Disponível em: <http://dx.doi.org/10.1007/s10479-004-5022-1>.

[65] HORST, R.AND THOAI, N. V. "DC Programming: Overview", *Journal of Optimization Theory and Applications*, v. 103, n. 1, pp. 1–43, 1999. ISSN: 1573-2878. doi: 10.1023/A:1021765131316. Disponível em: <http://dx.doi.org/10.1023/A:1021765131316>.

[66] HOAI AN, L. T., TAO, P. D., MUU, L. D. "A Combined D.C. Optimization—Ellipsoidal Branch-and-Bound Algorithm for Solving Nonconvex Quadratic Programming Problems", *Journal of Combinatorial Optimization*, v. 2, n. 1, pp. 9–28, 1998. ISSN: 1573-2886. doi: 10.1023/A:1009777410170. Disponível em: <http://dx.doi.org/10.1023/A:1009777410170>.

[67] BOMZE, I. M., LOCATELLI, M. "Undominated d.c. Decompositions of Quadratic Functions and Applications to Branch-and-Bound Approaches", *Computational Optimization and Applications*, v. 28, n. 2, pp. 227–245, 2004. ISSN: 1573-2894. doi: 10.1023/B:COAP.0000026886.61324.e4. Disponível em: <http://dx.doi.org/10.1023/B:COAP.0000026886.61324.e4>.

[68] ANSTREICHER, K. M., BURER, S. "D.C. Versus Copositive Bounds for Standard QP", *Journal of Global Optimization*, v. 33, n. 2, pp. 299–312, 2005. ISSN: 1573-2916. doi: 10.1007/s10898-004-4312-0. Disponível em: <http://dx.doi.org/10.1007/s10898-004-4312-0>.

[69] BOMZE, I. M., DÜR, M., DE KLERK, E., et al. "On Copositive Programming and Standard Quadratic Optimization Problems", *Journal of Global Optimization*, v. 18, n. 4, pp. 301–320, 2000. ISSN: 1573-2916. doi: 10.1023/A:1026583532263. Disponível em: <http://dx.doi.org/10.1023/A:1026583532263>.

[70] ZHENG, X. J., SUN, X. L., LI, D. "Nonconvex quadratically constrained quadratic programming: best D.C. decompositions and their SDP representations", *Journal of Global Optimization*, v. 50, n. 4, pp. 695–712, 2010. ISSN: 1573-2916. doi: 10.1007/s10898-010-9630-9. Disponível em: <http://dx.doi.org/10.1007/s10898-010-9630-9>.

[71] "GCC, the GNU Compiler Collection". Software. Disponível em: <https://gcc.gnu.org/>.

[72] ANGERSON, E., BAI, Z., DONGARRA, J., et al. "LAPACK: A portable linear algebra library for high-performance computers". In: *Supercomputing '90., Proceedings of*, pp. 2–11, Nov 1990. doi: 10.1109/SUPERC.1990.129995.

[73] "BLAS, Basic Linear Algebra Subprograms". Software. Disponível em: <http://www.netlib.org/blas/>.

[74] BELOTTI, P., LEE, J., LIBERTI, L., et al. "Branching and bounds tightening techniques for non-convex MINLP", *Optimization Methods and Software*, v. 24, n. 4-5, pp. 597–634, 2009. doi: 10.1080/10556780903087124. Disponível em: <http://dx.doi.org/10.1080/10556780903087124>.

[75] WIEGELE, A. "Biq Mac Library - Binary quadratic and Max cut Library". 2007. Disponível em: <http://biqmac.uni-klu.ac.at/biqmaclib.html>.

[76] BURER, S., VANDENBUSSCHE, D. "A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations", *Mathematical Programming*, v. 113, n. 2, pp. 259–282, 2008. ISSN: 1436-4646. doi: 10.1007/s10107-006-0080-6. Disponível em: <http://dx.doi.org/10.1007/s10107-006-0080-6>.

[77] GAMS. "GLOBAL Library". Disponível em: <http://www.gamsworld.org/global/globallib.htm>.

[78] CALAMAI, P. H., VICENTE, L. N., JÚDICE, J. J. "A new technique for generating quadratic programming test problems", *Mathematical Programming*, v. 61, n. 1, pp. 215–231, 1993. ISSN: 1436-4646. doi: 10.1007/BF01582148. Disponível em: <http://dx.doi.org/10.1007/BF01582148>.

[79] ALIZADEH, F. "Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization", *SIAM Journal on Optimization*, v. 5, pp. 13–51, 1993.

Apêndice A

Instâncias de teste de Programação Não Linear Inteira Mista adotadas

As 152 instâncias de teste de Programação Não Linear Inteira Mista binária utilizadas na Parte I deste trabalho foram obtidas de [43] no formato modelo AMPL compilado (arquivos .nl). Todas as instâncias foram rodados como problemas de minimização. A Tabela A.1 traz detalhes sobre cada instância. A coluna Obj traz o tipo da função objetivo (linear, quadrática ou não linear geral), a coluna n_x traz o número de variáveis contínuas, a coluna n_y traz o número de variáveis binárias enquanto a coluna m traz o total de restrições. m_{nl}, m_q e m_l são, respectivamente, o número de restrições não lineares gerais (não quadráticas), quadráticas e lineares. A coluna $nz_{\nabla g}$ traz o número de elementos não nulos na derivada primeira das restrições não lineares gerais (não quadráticas) e a coluna $nz_{\nabla H_{nl}}$ traz o número de não-zeros da parte não linear geral (não quadrática) da Hessiana do Lagrangeano.

Tabela A.1: Instâncias de teste de PNLIM

Instância	**Obj**	n_x	n_y	m	m_{nl}	m_q	m_l	$nz_{\nabla g}$	$nz_{\nabla H_{nl}}$
BatchS101006M	nl	149	129	1019	1	0	1018	10	79
BatchS121208M	nl	203	203	1511	1	0	1510	12	95
BatchS151208M	nl	242	203	1781	1	0	1780	15	98
BatchS201210M	nl	307	251	2327	1	0	2326	20	103
CLay0203H	linear	72	18	132	24	0	108	72	30
CLay0203M	linear	12	18	54	0	24	30	0	0
CLay0204H	linear	132	32	234	32	0	202	96	40
CLay0204M	linear	20	32	90	0	32	58	0	0
CLay0205H	linear	210	50	365	40	0	325	120	50
CLay0205M	linear	30	50	135	0	40	95	0	0
CLay0303H	linear	78	21	150	36	0	114	108	45
CLay0303M	linear	12	21	66	0	36	30	0	0
CLay0304H	linear	140	36	258	48	0	210	144	60
CLay0304M	linear	20	36	106	0	48	58	0	0
CLay0305H	linear	220	55	395	60	0	335	180	75
CLay0305M	linear	30	55	155	0	60	95	0	0
FLay02H	linear	42	4	51	2	0	49	4	2
FLay02M	linear	10	4	11	2	0	9	4	2

Continua na próxima página

Tabela A.1: Instâncias de teste de PNLIM (*continuação da página anterior*)

Instância	**Obj**	n_x	n_y	m	m_{nl}	m_q	m_l	$nz_{\nabla g}$	$nz_{\nabla H_{nl}}$
FLay03H	linear	110	12	144	3	0	141	6	3
FLay03M	linear	14	12	24	3	0	21	6	3
FLay04H	linear	210	24	282	4	0	278	8	4
FLay04M	linear	18	24	42	4	0	38	8	4
FLay05H	linear	342	40	465	5	0	460	10	5
FLay05M	linear	22	40	65	5	0	60	10	5
FLay06H	linear	506	60	693	6	0	687	12	6
FLay06M	linear	26	60	93	6	0	87	12	6
fo7_2	linear	72	42	211	14	0	197	28	14
fo7	linear	72	42	211	14	0	197	28	14
fo8	linear	90	56	273	16	0	257	32	16
fo9	linear	110	72	343	18	0	325	36	18
o7_2	linear	72	42	211	14	0	197	28	14
o7	linear	72	42	211	14	0	197	28	14
RSyn0805H	linear	424	296	1886	12	0	1874	36	12
RSyn0805M02H	linear	552	148	1045	6	0	1039	18	30
RSyn0805M02M	linear	212	148	769	6	0	763	18	6
RSyn0805M03H	linear	828	222	1698	9	0	1689	27	45
RSyn0805M03M	linear	318	222	1284	9	0	1275	27	9
RSyn0805M04H	linear	1104	296	2438	12	0	2426	36	60
RSyn0805M04M	linear	424	296	1886	12	0	1874	36	12
RSyn0805M	linear	101	69	286	3	0	283	9	3
RSyn0810H	linear	484	336	2140	24	0	2116	72	24
RSyn0810M02H	linear	622	168	1188	12	0	1176	36	60
RSyn0810M02M	linear	242	168	866	12	0	854	36	12
RSyn0810M03H	linear	933	252	1935	18	0	1917	54	90
RSyn0810M03M	linear	363	252	1452	18	0	1434	54	18
RSyn0810M04H	linear	1244	336	2784	24	0	2760	72	120
RSyn0810M04M	linear	484	336	2140	24	0	2116	72	24
RSyn0810M	linear	111	74	312	6	0	306	18	6
RSyn0815H	linear	564	376	2430	44	0	2386	132	44
RSyn0815M02H	linear	710	188	1361	22	0	1339	66	104
RSyn0815M02M	linear	282	188	981	22	0	959	66	22
RSyn0815M03H	linear	1065	282	2217	33	0	2184	99	156
RSyn0815M03M	linear	423	282	1647	33	0	1614	99	33
RSyn0815M04H	linear	1420	376	3190	44	0	3146	132	208
RSyn0815M04M	linear	564	376	2430	44	0	2386	132	44
RSyn0815M	linear	126	79	347	11	0	336	33	11
RSyn0820H	linear	604	416	2676	56	0	2620	168	56
RSyn0820M02H	linear	770	208	1500	28	0	1472	84	134
RSyn0820M02M	linear	302	208	1074	28	0	1046	84	28
RSyn0820M03H	linear	1155	312	2448	42	0	2406	126	201
RSyn0820M03M	linear	453	312	1809	42	0	1767	126	42
RSyn0820M04H	linear	1540	416	3528	56	0	3472	168	268
RSyn0820M04M	linear	604	416	2676	56	0	2620	168	56
RSyn0820M	linear	131	84	371	14	0	357	42	14
RSyn0830H	linear	744	496	3192	80	0	3112	240	80
RSyn0830M02H	linear	924	248	1794	40	0	1754	120	194
RSyn0830M02M	linear	372	248	1272	40	0	1232	120	40
RSyn0830M03H	linear	1386	372	2934	60	0	2874	180	291
RSyn0830M03M	linear	558	372	2151	60	0	2091	180	60
RSyn0830M04H	linear	1848	496	4236	80	0	4156	240	388
RSyn0830M04M	linear	744	496	3192	80	0	3112	240	80
RSyn0830M	linear	156	94	425	20	0	405	60	20
RSyn0840H	linear	864	576	3728	112	0	3616	336	112

Continua na próxima página

Tabela A.1: Instâncias de teste de PNLIM (*continuação da página anterior*)

Instância	**Obj**	n_x	n_y	m	m_{nl}	m_q	m_l	$nz_{\nabla g}$	$nz_{\nabla H_{nl}}$
RSyn0840M02H	linear	1072	288	2106	56	0	2050	168	268
RSyn0840M02M	linear	432	288	1480	56	0	1424	168	56
RSyn0840M03H	linear	1608	432	3447	84	0	3363	252	402
RSyn0840M03M	linear	648	432	2508	84	0	2424	252	84
RSyn0840M04H	linear	2144	576	4980	112	0	4868	336	536
RSyn0840M04M	linear	864	576	3728	112	0	3616	336	112
RSyn0840M	linear	176	104	484	28	0	456	84	28
SLay04H	quad	116	24	174	0	0	174	0	0
SLay04M	quad	20	24	54	0	0	54	0	0
SLay05H	quad	190	40	290	0	0	290	0	0
SLay05M	quad	30	40	90	0	0	90	0	0
SLay06H	quad	282	60	435	0	0	435	0	0
SLay06M	quad	42	60	135	0	0	135	0	0
SLay07H	quad	392	84	609	0	0	609	0	0
SLay07M	quad	56	84	189	0	0	189	0	0
SLay08H	quad	520	112	812	0	0	812	0	0
SLay08M	quad	72	112	252	0	0	252	0	0
SLay09H	quad	666	144	1044	0	0	1044	0	0
SLay09M	quad	90	144	324	0	0	324	0	0
SLay10H	quad	830	180	1305	0	0	1305	0	0
SLay10M	quad	110	180	405	0	0	405	0	0
Syn05H	linear	37	5	58	3	0	55	9	15
Syn05M02H	linear	84	20	151	6	0	145	18	30
Syn05M02M	linear	40	20	101	6	0	95	18	6
Syn05M03H	linear	126	30	249	9	0	240	27	45
Syn05M03M	linear	60	30	174	9	0	165	27	9
Syn05M04H	linear	168	40	362	12	0	350	36	60
Syn05M04M	linear	80	40	262	12	0	250	36	12
Syn05M	linear	15	5	28	3	0	25	9	3
Syn10H	linear	67	10	112	6	0	106	18	30
Syn10M02H	linear	154	40	294	12	0	282	36	60
Syn10M02M	linear	70	40	198	12	0	186	36	12
Syn10M03H	linear	231	60	486	18	0	468	54	90
Syn10M03M	linear	105	60	342	18	0	324	54	18
Syn10M04H	linear	308	80	708	24	0	684	72	120
Syn10M04M	linear	140	80	516	24	0	492	72	24
Syn10M	linear	25	10	54	6	0	48	18	6
Syn15H	linear	106	15	181	11	0	170	33	52
Syn15M02H	linear	242	60	467	22	0	445	66	104
Syn15M02M	linear	110	60	313	22	0	291	66	22
Syn15M03H	linear	363	90	768	33	0	735	99	156
Syn15M03M	linear	165	90	537	33	0	504	99	33
Syn15M04H	linear	484	120	1114	44	0	1070	132	208
Syn15M04M	linear	220	120	806	44	0	762	132	44
Syn15M	linear	40	15	89	11	0	78	33	11
Syn20H	linear	131	20	233	14	0	219	42	67
Syn20M02H	linear	302	80	606	28	0	578	84	134
Syn20M02M	linear	130	80	406	28	0	378	84	28
Syn20M03H	linear	453	120	999	42	0	957	126	201
Syn20M03M	linear	195	120	699	42	0	657	126	42
Syn20M04H	linear	604	160	1452	56	0	1396	168	268
Syn20M04M	linear	260	160	1052	56	0	996	168	56
Syn20M	linear	45	20	113	14	0	99	42	14
Syn30H	linear	198	30	345	20	0	325	60	97
Syn30M02H	linear	456	120	900	40	0	860	120	194

Continua na próxima página

Tabela A.1: Instâncias de teste de PNLIM (*continuação da página anterior*)

Instância	**Obj**	n_x	n_y	m	m_{nl}	m_q	m_l	$nz_{\nabla g}$	$nz_{\nabla H_{nl}}$
Syn30M02M	linear	200	120	604	40	0	564	120	40
Syn30M03H	linear	684	180	1485	60	0	1425	180	291
Syn30M03M	linear	300	180	1041	60	0	981	180	60
Syn30M04H	linear	912	240	2160	80	0	2080	240	388
Syn30M04M	linear	400	240	1568	80	0	1488	240	80
Syn30M	linear	70	30	167	20	0	147	60	20
Syn40H	linear	262	40	466	28	0	438	84	134
Syn40M02H	linear	604	160	1212	56	0	1156	168	268
Syn40M02M	linear	260	160	812	56	0	756	168	56
Syn40M03H	linear	906	240	1998	84	0	1914	252	402
Syn40M03M	linear	390	240	1398	84	0	1314	252	84
Syn40M04H	linear	1208	320	2904	112	0	2792	336	536
Syn40M04M	linear	520	320	2104	112	0	1992	336	112
Syn40M	linear	90	40	226	28	0	198	84	28
trimloss12	linear	156	644	372	12	0	360	3780	300
trimloss2	linear	6	31	24	2	0	22	52	10
trimloss4	linear	20	85	64	4	0	60	176	36
trimloss5	linear	30	131	90	5	0	85	280	55
trimloss6	linear	56	289	154	7	0	147	770	105
trimloss7	linear	56	289	154	7	0	147	770	105
Water0202	quad	106704	7	107209	0	0	107209	0	0
Water0202R	quad	188	7	283	0	0	283	0	0
Water0303	quad	107208	14	108217	0	0	108217	0	0
Water0303R	quad	370	14	556	0	0	556	0	0

Apêndice B

Demonstração do Teorema 7.1

Fornecemos aqui, a demonstração do Teorema 7.1 a seguir:

Teorema 7.1. *Para quaisquer parâmetros m_i, $i = 1, \ldots, k$, tais que $m_1 \geq m_2 \geq \cdots m_k > 0$, a decomposição por autovalores de Q resolve o problema de minimização de soma ponderada de autovalores $(PSPAV)$.*

Lema B.1. *$(PSPAV)$ é equivalente ao problema de PSD:*

$$
\begin{array}{lll}
(PPSD) & min & \sum_{i=1}^{k} i z_i + \sum_{i=1}^{k} Tr(V_i) \ , \\
 & sujeito\ a: & z_i I + V_i - (m_i - m_{i+1}) R \succeq 0 \ , \quad i = 1,2,\ldots,k \ ; \\
 & & Q + R \succeq 0 \ ; \\
 & & R \succeq 0 \ ; \\
 & & V_i \succeq 0 \ , \quad i = 1,2,\ldots,k \ .
\end{array}
$$

e seu dual pode ser escrito como

$$
\begin{array}{lll}
(DPSD) & max & Q \bullet X \ , \\
 & sujeito\ a: & Tr(Y_i) = i \ , \quad i = 1,2,\ldots,k \ ; \\
 & & \sum_{i=1}^{k} (m_i - m_{i+1}) Y_i + X \succeq 0 \ ; \\
 & & 0 \preceq Y_i \preceq I \ , \quad i = 1,2,\ldots,k \ ; \\
 & & X \preceq 0 \ .
\end{array}
$$

Demonstração. Aplicando a descrição em [79], $(PSPAV)$ é um problema de minimização convexa e pode ser reformulado como $(PPSD)$, ou, equivalentemente

$$
\begin{array}{lll}
(PPSD') & \min & \sum_{i=1}^{k} i(\bar{z}_i - \underline{z}_i) + \sum_{i=1}^{k} \mathrm{Tr}(V_i) \ , \\
& \text{sujeito a:} & (\bar{z}_i - \underline{z}_i)I + V_i - (m_i - m_{i+1})R - W_i = 0 \ , \quad i = 1,2,\ldots,k \ ; \\
& & Q + R - S = 0 \ ; \\
& & R, S \succeq 0 \ ; \\
& & V_i, W_i \succeq 0 \ , \quad i = 1,2,\ldots,k \ ; \\
& & \bar{z}_i, \ \underline{z}_i \geq 0 \ , \quad i = 1,2,\ldots,k \ .
\end{array}
$$

É conveniente escrever $(PPSD')$ na forma padrão:

$$
\begin{array}{lll}
(SPPSD) & \min & F_0 \bullet \Gamma \ , \\
& \text{sujeito a:} & F_j \bullet \Gamma = c_j \ , \quad j = 1,2,\ldots,m \ ; \\
& & \Gamma \succeq 0 \ ,
\end{array}
$$

onde " $\bullet$ " é o produto interno matricial usual:

$$
A \bullet B := \sum_{i,j} A_{ij} B_{ij} = \mathrm{Tr}(A'B).
$$

Defina

$$
\Gamma := \mathrm{Diag}\left(\bar{Z},\underline{Z},S,R,W_1,W_2,\ldots,W_k,V_1,V_2,\ldots,V_k\right),
$$

onde $\bar{Z}$ é uma matriz diagonal com $\bar{Z}_{ii} = \bar{z}_i$, para $i = 1,\ldots,k$ (uma definição análoga se aplica para $\underline{Z}$), e

$$
F_0 := \mathrm{Diag}\left(G, -G,0,0,0,0,\ldots,0,I,I,\ldots,I\right),
$$

onde G é uma matriz diagonal, com $G_{jj} = j$, para $j = 1,\ldots,k$. A função objetivo do problema $(PPSD')$ pode então ser escrita como $F_0 \bullet \Gamma$.

Definimos agora $k+1$ grupos de $n(n+1)/2$ matrizes. Cada grupo é usado para formular uma restrição de igualdade do problema $(PPSD')$. As primeiras n matrizes de cada grupo são usadas para determinar os valores diagonais das matrizes no lado esquerdo das restrições e as outras $n(n-1)/2$ matrizes do grupo são usadas para determinar os valores não diagonais. A notação E_i é usada a seguir para representar a matriz diagonal com elemento (i,i) igual a um e todos os outros iguais a zero. Adicionalmente, $E_{\gamma\beta}$ denota a matriz simétrica com elementos (γ,β) e (β,γ) iguais a um e todos os outros iguais a zero.

Para a primeira restrição em $(PPSD')$,

$$
(\bar{z}_1 - \underline{z}_1)I + V_1 - (m_1 - m_2)R - W_1 = 0 \ ,
$$

definimos

$$F_j := \text{Diag}\,(E_1, -E_1, 0, -\tilde{m}_1 E_j, -E_j, 0, \ldots, 0, E_j, 0, \ldots, 0)\,,$$

para $j = 1, \ldots, n$, e

$$F_{n+j} := \frac{1}{2}\text{Diag}\,(0, 0, 0, -\tilde{m}_1 E_{\gamma\beta}, -E_{\gamma\beta}, 0, \ldots, 0, E_{\gamma\beta}, 0, \ldots, 0)\,,$$

para $j = 1, \ldots, n(n-1)/2$, onde $\tilde{m}_1 := m_1 - m_2$. Os índices γ, β satisfazem $\gamma, \beta \in \{1, \ldots, n\}$, com $\gamma > \beta$. Cada par (γ, β) corresponde unicamente a um índice j.

A restrição $(\bar{z}_1 - \underline{z}_1)I + V_1 - (m_1 - m_2)R - W_1 = 0$ pode então ser escrita como $F_j \bullet \Gamma = 0$, para $j = 1, \ldots, n(n+1)/2$.

Um grupo análogo de matrizes é usado para formular cada uma das k primeiras restrições de igualdade de $(PPSD')$. Para a última restrição,

$$(\bar{z}_k - \underline{z}_k)I + V_k - (m_k - m_{k+1})R - W_k = 0\ ,$$

definimos

$$F_{\nu_1+j} := \text{Diag}\,(E_k, -E_k, 0, -\tilde{m}_k E_j, 0, 0, \ldots, -E_j, 0, 0, \ldots, E_j)\,,$$

para $j = 1, \ldots, n$, e

$$F_{\nu_1+n+j} := \frac{1}{2}\text{Diag}\,(0, 0, 0, -\tilde{m}_k E_{\gamma\beta}, 0, 0, \ldots, -E_{\gamma\beta}, 0, 0, \ldots, E_{\gamma\beta})\,,$$

para $j = 1, \ldots, n(n-1)/2$, onde $\tilde{m}_k := m_k - m_{k+1}$ e ν_1 é o número de matrizes usadas para formular as $k-1$ primeiras restrições.

Por fim, para formular a restrição $Q + R - S = 0$, definimos:

$$F_{\nu_2+j} := \text{Diag}\,(0, 0, E_j, -E_j, 0, 0, \ldots, 0, 0, 0, 0, \ldots, 0)$$

e $c_{\nu_2+j} := Q_{jj}$, para $j = 1, \ldots, n$, e

$$F_{\nu_2+n+j} := \frac{1}{2}\text{Diag}\,(0, 0, E_{\gamma\beta}, -E_{\gamma\beta}, 0, 0, \ldots, 0, 0, 0, 0, \ldots, 0)$$

e $c_{\nu_2+n+j} := Q_{\gamma\beta}$, para $j = 1, \ldots, n(n-1)/2$, onde ν_2 é o número de matrizes usadas para formular as k primeiras restrições.

O dual de $(SPPSD)$ é

$$\begin{array}{lll} (SDPSD) & \max & c'\delta\ , \\ & \text{sujeito a:} & F(\delta) \succeq 0\ , \end{array} \tag{B.1}$$

onde

$$F(\delta) := F_0 - \sum_{i=1}^{m} \delta_i F_i \ .$$

Considerando as definições acima para F_0 e F_j, $j = 1,\ldots,m$, onde $m = (k+1) \times (n(n+1)/2)$, temos:

$$F(\delta) = \text{Diag}\left(U, -U, -X, \tilde{X}, Y_1, Y_2, \ldots, Y_k, I - Y_1, I - Y_2, \ldots, I - Y_k\right),$$

onde U é uma matriz diagonal com $U_{ii} = i - \text{Tr}(Y_i)$, para $i = 1,\ldots,k$, e $\tilde{X} := \sum_{i=1}^{k}(m_i - m_{i+1})Y_i + X$.

O problema dual $(SDPSD)$ pode então ser escrito na forma equivalente $(DPSD)$. □

Lema B.2. *Sejam R^*, V_i^*, e z_i^*, para $i = 1,2,\ldots,k$, uma solução viável para o problema primal $(PPSD)$. Sejam X^* e Y_i^*, para $i = 1,\ldots,k$, uma solução viável para o problema dual $(DPSD)$. Então, as soluções são primal e dual ótimas, se, e, somente se, elas satisfazem as restrições de complementaridade:*

$$\left(\sum_{i=1}^{k}(m_i - m_{i+1})Y_i^* + X^*\right) R^* = 0 \ ; \tag{B.2}$$

$$(I - Y_i^*)V_i^* = 0 \ , \quad i = 1,2,\ldots,k \ ; \tag{B.3}$$

$$\left(z_i^* I + V_i^* - (m_i - m_{i+1})R^*\right) Y_i^* = 0 \ , \quad i = 1,2,\ldots,k \ ; \tag{B.4}$$

$$(Q + R^*)X^* = 0 \ . \tag{B.5}$$

Demonstração. As condições de complementaridade para os pares primal e dual padrão $(SPPSD)$ e $(SDPSD)$ são dados por $F(\delta)\Gamma = 0$. Usando as expressões para $F(\delta)$ e Γ da prova do lema anterior, obtemos as condições de otimalidade para problemas $(PPSD)$ e $(DPSD)$. □

Lema B.3. *Seja*

$$\sum_{i=1}^{n} \lambda_i v_i v_i' = \sum_{i \in P} \lambda_i v_i v_i' - \sum_{i \in N}(-\lambda_i) v_i v_i'$$

a decomposição por autovalores de Q , onde $\lambda_i > 0$ para $i \in P$ e $\lambda_i < 0$ para $i \in N$. Então $Q = \tilde{P} - \tilde{R}$,onde $\tilde{P} := \sum_{i \in P} \lambda_i v_i v_i'$ e $\tilde{R} := \sum_{i \in N}(-\lambda_i) v_i v_i'$.

Sem perda de generalidade, seja $N = \{1,\ldots,\bar{n}\}$ e $(-\lambda_1) \geq (-\lambda_2) \geq \ldots \geq (-\lambda_{\bar{n}})$.

Seja

$$\begin{aligned}
&\bar{R} = \tilde{R} \ ; \\
&\bar{z}_i = (m_i - m_{i+1})(-\lambda_{i+1}) \ , \quad i = 1,2,\ldots,\bar{n}-1 \ ; \\
&\bar{z}_i = 0 \ , \quad i = \bar{n},\bar{n}+1,\ldots,k \ ; \\
&\bar{V}_i = \sum_{j=1}^{\underline{i}} \left((m_i - m_{i+1})(-\lambda_j) - \bar{z}_i\right) v_j v_j' \ , \quad i = 1,2,\ldots,k \ ;
\end{aligned}$$

onde $\underline{i} = \min\{i,\bar{n}\}$, *e*

$$\begin{aligned}
&\bar{Y}_i = \sum_{j=1}^{i} v_j v_j' \ , \quad i = 1,2,\ldots,k \ ; \\
&\bar{X} = -\sum_{i=1}^{k} (m_i - m_{i+1})\bar{Y}_i \ .
\end{aligned}$$

Então $\bar{R}$, $\bar{V}_i$, *e* $\bar{z}_i$, *para* $i = 1,2,\ldots,k$, *é uma solução ótima para* $(PPSD)$ *e,* $\bar{X}$ *e* $\bar{Y}_i$, *para* $i = 1,\ldots,k$, *é uma solução ótima para* $(DPSD)$.

Demonstração. Primeiro, verificaremos que a solução dada por $\bar{R}$, $\bar{V}_i$, e $\bar{z}_i$, para $i = 1,2,\ldots,k$, é viável para o problema $(PPSD)$.

Claramente $\bar{R} \succeq 0$. A semidefinição positiva das matrizes $\bar{V}_i$ resulta da ordenação dos autovalores de $\bar{R}$, $(-\lambda_1) \geq (-\lambda_2) \geq \ldots \geq (-\lambda_{\bar{n}}) > 0$. Uma vez que $Q + \bar{R} = \tilde{P}$, temos ainda $Q + \bar{R} \succeq 0$.

No que diz respeito às restrições $z_i I + V_i - (m_i - m_{i+1})R \succeq 0 \ , \quad i = 1,2,\ldots,k$, dividimos a análise em dois casos. Para $i < \bar{n}$, temos

$$\begin{aligned}
\bar{z}_i I \ + \ & \bar{V}_i - (m_i - m_{i+1})\bar{R} \\
= \ & (m_i - m_{i+1})(-\lambda_{i+1})I + \textstyle\sum_{j=1}^{i} \left((m_i - m_{i+1})(-\lambda_j) - \bar{z}_i\right) v_j v_j' \\
& -(m_i - m_{i+1}) \textstyle\sum_{j=1}^{\bar{n}} (-\lambda_j) v_j v_j' \\
= \ & (m_i - m_{i+1})(-\lambda_{i+1})I - \textstyle\sum_{j=1}^{i} \bar{z}_i v_j v_j' - (m_i - m_{i+1}) \sum_{j=i+1}^{\bar{n}} (-\lambda_j) v_j v_j' \\
= \ & (m_i - m_{i+1}) \left((-\lambda_{i+1})I - \textstyle\sum_{j=1}^{i} (-\lambda_{i+1}) v_j v_j' - \sum_{j=i+1}^{\bar{n}} (-\lambda_j) v_j v_j' \right) .
\end{aligned} \tag{B.6}$$

Portanto, a semidefinição positiva de $\bar{z}_i I + \bar{V}_i - (m_i - m_{i+1})\bar{R}$ também resulta da ordenação dos autovalores de $\bar{R}$.

Quando $i \geq \bar{n}$, $\bar{z}_i = 0$ e $\bar{V}_i = (m_i - m_{i+1})\bar{R}_0$. As restrições são então claramente satisfeitas.

É fácil verificar que a solução dada por $\bar{X}$ e $\bar{Y}_i$, para $i = 1,\ldots,k$, é viável para $(DPSD)$.

Verificamos finalmente que as soluções satisfazem as restrições de complementaridade. A restrição (B.2) é satisfeita a partir da definição de $\bar{X}$. As restrições (B.3)

são satisfeitas porque a ortonormalidade dos autovetores de $\bar{R}$ resulta em $\bar{V}_i = \bar{Y}_i \bar{V}_i$.

No que diz respeito às restrições (B.4), novamente dividimos a análise em dois casos. Para $i < \bar{n}$, usando a última expressão em (B.6), temos:

$$\begin{aligned}(\bar{z}_i I + \bar{V}_i - (m_i - m_{i+1})\bar{R})\bar{Y}_i &= (m_i - m_{i+1})\left((-\lambda_{i+1})I \right. \\ &- \textstyle\sum_{j=1}^{i}(-\lambda_{i+1})v_j v_j' \\ &- \left. \textstyle\sum_{j=i+1}^{\bar{n}}(-\lambda_j)v_j v_j\right)\left(\sum_{j=1}^{i} v_j v_j\right) \\ &= 0.\end{aligned}$$

A última igualdade também resulta da ortonormalidade dos autovetores de $\bar{R}$.

Quando $i \geq \bar{n}$, $\bar{z}_i = 0$ e $\bar{V}_i = (m_i - m_{i+1})\bar{R}$. As restrições (B.4) são então claramente satisfeitas.

A última restrição (B.5) é satisfeita porque os autovetores de $\tilde{P} = Q + \bar{R}$ são ortogonais aos autovetores de $\bar{X}$. □

Teorema 7.1 segue agora.

Apêndice C

Instâncias de teste de Programação Quadrática Não Convexa Selecionadas Para Comparação Com *Solvers*

Listamos aqui as instâncias de teste de programação quadrática não convexa utilizadas para comparação de nossa abordagem desenvolvida na Parte II com *solvers* conhecidos. A Tabela C.1, apresentas detalhes das 10% mais difíceis instâncias para nossa abordagem que foram tomadas na comparação dos *solvers* realizada na Subseção 8.1.3. A coluna n traz o número de variáveis da instância, ao passo que a coluna n_I exibe o número de variáveis inteiras e a coluna m_l discrimina o número de restrições lineares. Por fim, a coluna nz_Q traz o número de elementos não zeros na matriz Q_0.

Tabela C.1: Instâncias de teste de PQ não convexa

Instância	n	n_I	ml	nz_Q
R-BiqMac				
gka10b	125	0	0	7788
gka9b	100	0	0	5002
pm1d_100.0	100	0	0	4901
pm1d_100.1	100	0	0	4901
pm1d_100.4	100	0	0	4901
pm1d_100.6	100	0	0	4901
pm1d_100.7	100	0	0	4901
pm1d_80.0	80	0	0	3128
pm1d_80.1	80	0	0	3128
pm1d_80.4	80	0	0	3128
pm1d_80.5	80	0	0	3128
pm1d_80.6	80	0	0	3128
pm1d_80.7	80	0	0	3128
pm1s_100.3	100	0	0	495
pm1s_100.7	100	0	0	495

Continua na próxima página

Tabela C.1: Instâncias de teste de PQ não convexa (*continuação da página anterior*)

Instância	n	n_I	ml	nz_Q
pm1s_80.0	80	0	0	316
pm1s_80.4	80	0	0	316
pm1s_80.6	80	0	0	316
pm1s_80.9	80	0	0	316
t2g20_5555	400	0	0	800
t3g6_5555	216	0	0	648
t3g7_5555	343	0	0	1029
t3g7_7777	343	0	0	1029
w01_100.0	100	0	0	495
w01_100.6	100	0	0	495
w05_100.4	100	0	0	2475
w05_100.5	100	0	0	2475
w05_100.6	100	0	0	2475
w05_100.8	100	0	0	2475
w09_100.0	100	0	0	4455
w09_100.1	100	0	0	4455
w09_100.3	100	0	0	4455
w09_100.4	100	0	0	4455
w09_100.8	100	0	0	4455
w09_100.9	100	0	0	4455
BoxQP/MIBoxQP				
spar100-050-1	100	0/70	0	5050
spar125-025-1	125	0/62	0	7875
spar125-025-2	125	0/62	0	7875
spar125-025-3	125	0/62	0	7875
spar125-050-1	125	0/62	0	7875
spar125-050-2	125	0/62	0	7875
spar125-050-3	125	0/62	0	7875
spar125-075-1	125	0/62	0	7875
spar125-075-2	125	0/62	0	7875
spar125-075-3	125	0/62	0	7875
GLOBALLIB				
ex2_1_10	20	0	10	20
ex2_1_8	24	0	10	24
qp1	50	0	2	1275
qp2	50	0	2	1275
qp4	79	0	31	29
st_m2	30	0	21	30
st_rv7	30	0	20	30
st_rv8	40	0	20	40
st_rv9	50	0	20	50
Random				
myqp1_060_g2p4_00_15_15	60	0	90	692
myqp1_080_g3p2_13_13_14	80	0	120	1085
myqp1_080_g3p3_13_13_14	80	0	120	1166
myqp1_100_g2p1_00_25_25	100	0	150	1799
myqp1_100_g2p2_00_25_25	100	0	150	2046
myqp1_100_g2p4_00_25_25	100	0	150	1999
myqp1_100_g3p4_16_16_18	100	0	150	1756