O ALGORITMO DO VOLUME: CONVERGÊNCIA E RESOLUÇÃO DO PROBLEMA DE STEINER EM GRAFOS

Laura Silvia Bahiense da Silva Leite

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS GRADUAÇÃO DE ENGENHARIA 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.

Aprovada por:

Prof. Nelson Maculan Filho, Docteur Habilité.

Dr. Francisco Barahona, Docteur.

Profª Claudia A. Sagastizábal, Docteur Habilité.

Prof. Celso da C. Carneiro Ribeiro, Docteur Habilité.

Dr. Mário Veiga Ferraz Pereira, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 2000

LEITE, LAURA SILVIA BAHIENSE DA SILVA

O Algoritmo do Volume: Convergência e Resolução do Problema de Steiner em Grafos [Rio de Janeiro] 2000

VII, 70 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2000)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Otimização

2. Algoritmo do Volume

I. COPPE/UFRJ II. Título (série)

Agradecimentos

O trabalho de pesquisa para esta tese de doutorado foi realizado na Universidade Federal do Rio de Janeiro e na Pontifícia Universidade Católica do Rio de Janeiro. Agradeço pelo carinho com que fui acolhida em ambas instituições, em especial aos professores Nelson Maculan Filho (COPPE-Sistemas/UFRJ), Oscar Porto (DEE/PUC-Rio) e Marcus Vinícius Poggi de Aragão (DI/PUC-Rio).

Agradeço ao CNPq pela bolsa concedida para a realização deste trabalho. Agradeço à COPPE e ao CNPq pelo apoio financeiro que tornou possível a divulgação deste trabalho em congressos no Brasil e no exterior.

Sou grata ao professor Dr. Nelson Maculan Filho pelo interesse e dedicação com que acompanhou meu trabalho de pesquisa. Aproveito este espaço para fazer minhas as palavras de muitos, ressaltando seu grande coração, sua amizade e sua dedicação aos alunos e ao ensino de uma maneira geral. Uma pessoa realmente admirável com quem tive a honra de trabalhar.

Agradeço ao Dr. Francisco Barahona por ter proposto este trabalho e por seu apoio e sua orientação, mesmo à distância. Sou grata a Alicja Barahona por sua amizade e carinho. Aproveito para agradecer também ao *T.J. Watson Research Center* da IBM em *New York/USA* pelo estágio durante o mês de outubro de 1999.

Sou grata à professora Dra. Claudia Sagastizábal pela orientação em toda a primeira parte deste trabalho.

Agradeço a Oscar Porto por sua dedicação, seu companheirismo e sua amizade durante os últimos quatro anos. Seu carinho, sua compreensão e seu apoio foram fundamentais para que eu tivesse força para continuar.

Agradeço ao professor Dr. Celso Carneiro Ribeiro e ao Dr. Mário Veiga Ferraz Pereira pela participação na banca examinadora e pelo interesse neste trabalho. Agradeço em particular ao Dr. Mário Veiga por seu incentivo.

Agradeço finalmente a todos os meus amigos e colegas que sempre estiveram torcendo por mim e ajudando no que fosse possível. Em especial a Maria Clícia Sterling de Castro, Marcos de Mendonça Passini, Flávio Montenegro e Marcio de Moraes Palmeira.

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

O ALGORITMO DO VOLUME: CONVERGÊNCIA E RESOLUÇÃO DO PROBLEMA DE STEINER EM GRAFOS

Laura Silvia Bahiense da Silva Leite

Março/2000

Orientador: Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

Este trabalho é dividido em duas partes. Na Parte I propõe-se uma versão revisada do algoritmo do volume (VA) para programação linear, e apresenta-se uma prova de sua convergência. Originalmente, VA foi apresentado como um método de subgradientes para a resolução do problema dual, produzindo passos verdes/amarelos/vermelhos similares aos passos sérios/nulos dos métodos de feixes. Neste trabalho mostra-se que VA é na realidade um método de extragradientes. Junto com isso apresenta-se um novo padrão algorítmico baseado em extragradientes para a resolução de problemas de otimização não-suaves. A versão revisada do algoritmo do volume (RVA) apresentada neste trabalho introduz uma medida precisa da melhora mínima requerida na função dual para a declaração de um passo sério; esse é o ponto chave da prova de convergência. Na Parte II, propõe-se dois novos algoritmos para a resolução do problema de Steiner em grafos (SPG), baseados em VA e RVA. A relaxação contínua de uma formulação multi-fluxos do SPG é considerada, e então resolvida num contexto de relaxação Lagrangeana. Os algoritmos propostos não só produzem boas soluções duais como também estimam soluções primais da relaxação contínua do SPG que podem violar as restrições em no máximo 0.1%. Além disso, são propostas novas heurísticas primais baseadas nestas soluções primais estimadas. A boa qualidade dos limites inferiores e superiores obtidos permitiu resolver de maneira provadamente ótima muitas instâncias difíceis da literatura, sem pré-processamento e sem recorrer a nenhum método enumerativo. Além disso, os saltos de dualidade obtidos em instâncias para as quais não foi possível provar otimalidade foram bastante pequenos. Finalmente, reportam-se resultados computacionais para um grande número de problemas da literatura, incluídos aqueles contidos na biblioteca *SteinLib*.

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

THE VOLUME ALGORITHM: CONVERGENCE AND SOLVING THE STEINER TREE PROBLEM IN GRAPHS

Laura Silvia Bahiense da Silva Leite

March/2000

Advisor: Nelson Maculan Filho

Department: Sistems and Computer Science Engineering

This work is divided in two parts. In Part I a revised version of the volume algorithm (VA) for linear programming is proposed, and a proof of convergence is presented. Originally, VA was presented as a subgradient-like method for solving the dual, producing green/yellow/red steps similar to the serious/null steps of bundle methods. In this work it is shown that VA is in fact an extragradient method. Together with this it is presented a new algorithmic pattern based on extragradients for solving non-smooth optimization problems. The revised volume algorithm (RVA) presented in this work introduces a precise measure of the minimum improvement required on the dual function to declare a serious step; this is the key point in the proof of convergence. In Part II, two new algorithms for solving the Steiner Tree Problem in Graphs (SPG) are proposed, based on VA and RVA. The continuous relaxation of a multicommodity flow formulation of the SPG is considered and then solved in a context of Lagrangian relaxation. The proposed algorithms not only produce good dual solutions but also estimate primal solutions of the continuous relaxation of the SPG that might violate the constraints by at most 0.1%. Additionally, new primal heuristics based on those estimated primal solutions are also proposed. The good quality of the lower and upper bounds obtained made possible solving many difficult instances from the literature to proven optimality without preprocessing and without resorting to branching. Futhermore, the gaps obtained in instances for wich optimality could not be proven, have been fairly small. Finally, computational results are reported for a number of problems drown from the literature, including the ones contained in the *SteinLib* library.

Índice

Prefácio

Prefácio

A motivação fundamental deste trabalho foi o surgimento do *Algoritmo do Volume*, desenvolvido por Barahona e Anbil [5] em 1998, para a resolução de problemas de programação linear de grande porte. A elaboração deste algoritmo, segundo os autores, foi motivada pela falta de boa informação primal durante a resolução dos problemas duais em um contexto de relaxação Lagrangeana.

Uma das técnicas utilizadas para a resolução de problemas de programação linear de grande porte é a decomposição de Dantzig-Wolfe [26], que consiste em trabalhar com subconjuntos das colunas do problema original ou "mestre", que vão sendo geradas conforme necessário. Mais especificamente, cada nova coluna é obtida resolvendo-se um problema de programação linear chamado de subproblema ou "problema escravo". Quando o número de colunas a gerar é muito grande, este procedimento pode ficar lento demais. O algoritmo do volume pode ser visto como uma maneira rápida de aproximar a decomposição de Dantzig-Wolfe, mantendo a simplicidade do método de subgradientes, porém com critérios de parada bem definidos.

O nome "volume" é devido à maneira como as variáveis primais são calculadas. O *Teorema do Volume* [5, § 2, Teorema 2.2] prova que as variáveis primais do problema mestre da decomposição de Dantzig-Wolfe são proporcionais aos volumes definidos pelas faces que estão ativas em uma vizinhança de uma solução (ótima) do problema dual associado.

As soluções primais geradas pelo algoritmo do volume podem ser usadas ou como soluções aproximadas do problema original, ou como ponto de partida para métodos mais exatos (Simplex, por exemplo), ou, em caso de problemas de otimização combinatória, como informação primal da relaxação contínua do problema original para heurísticas que gerem soluções inteiras.

Os excelentes resultados computacionais apresentados em [5, 6, 7, 8] para problemas de *set partitioning, set covering, max-cut* e *facility location* validam a utilização do algoritmo do volume para problemas de otimização combinatória.

Na realidade, a ênfase de [5, 6, 7, 8] é mais em resultados numéricos; as propriedades de convergência do algoritmo não são analisadas. Isso motivou os estudos iniciais deste trabalho, proposto pelo Dr. Francisco Barahona durante sua visita ao Laboratório Nacional de Computação Científica (LNCC) no Rio de Janeiro em novembro de 1997, quando ele apresentou o algoritmo do volume. Os estudos sobre a convergência começaram durante a visita que fizemos ao *Institut National de Recherche en Informatique et Automatique (INRIA), Rocquencourt, France*, em julho de 1998, a convite da Dra. Claudia A. Sagastizábal; e continuaram no Rio de Janeiro por mais um

ano, orientados pela Dra. Claudia A. Sagastizábal e pelo Prof. Dr. Nelson Maculan Filho.

Em § 1.4.2 introduz-se uma *versão revisada do algoritmo do volume*, que difere ligeiramente do algoritmo original e para a qual prova-se convergência e apresenta-se um limite *a posteriori* para o erro de aproximação das soluções estimadas produzidas. Ao longo da prova de convergência do algoritmo do volume revisado, mostra-se que o algoritmo do volume na verdade não se encaixa na classe dos métodos de subgradientes, como pensavam originalmente seus autores, sendo melhor classificado como um método de extragradientes.

Uma interessante conseqüência da intenção inicial de provar a convergência do algoritmo do volume foi o desenvolvimento de um novo padrão algorítmico para problemas de otimização não-suaves (Non Smooth Optimization, NSO). Uma vez que sua principal característica consiste em atualizar os multiplicadores de Lagrange através de um deslocamento de ε-extragradiente, este método, introduzido em § 1.4.1, é chamado de ε-ESG.

Paralelamente foi sugerida pelo Dr. Francisco Barahona a elaboração de um algoritmo para a resolução do *Problema de Steiner em Grafos, SPG*, baseado no algoritmo do volume. A escolha deste problema foi devida fundamentalmente aos seguintes motivos: o SPG é um problema difícil, bastante estudado pela comunidade científica, e foi abordado por diversas técnicas, o que permitiria uma comparação rica com o novo algoritmo a ser desenvolvido; várias aplicações importantes do SPG, em particular a que diz respeito a circuitos eletrônicos VLSI, contribuíram nos últimos anos para aumentar o volume de pesquisa na procura de bons algoritmos capazes de resolver instâncias reais de grande dificuldade. Em § 2 apresenta-se o algoritmo desenvolvido em conjunto com o Dr. Francisco Barahona. Além dos limites inferiores produzidos pelo algoritmo proposto, foram desenvolvidas também três heurísticas primais para gerar limites superiores para SPG. Essas heurísticas são apresentadas em § 2.3.

É importante ressaltar que o trabalho de pesquisa para o desenvolvimento do algoritmo para a resolução do SPG foi viabilizado graças à visita do Dr. Francisco Barahona à COPPE-Sistemas e Computação em dezembro de 1998, ao estágio que fiz no *T. J. Watson Research Center* da *IBM* em *Yorktown Heights, NY, USA* em outubro de 1999, e ao contato regular via *e-mail* e telefone com o Dr. Barahona.

Os resultados obtidos nesta Tese originaram dois trabalhos (*papers*) [3, 4]; o primeiro deles foi submetido para publicação na revista *Mathematical Programming* e o segundo encontra-se em fase final de redação e será submetido para publicação na revista *Journal on Combinatorial Optimization.*

Resultados parciais relativos ao algoritmo desenvolvido para a resolução do SPG foram apresentados na 6ª *Escuela Latino Americana de Verano de Investigación Operativa*, VI ELAVIO, realizada em Mendes, Rio de Janeiro, em janeiro de 1999. O algoritmo do volume revisado foi apresentado no *Third Workshop on Applied Combinatorial Optimization*, que teve lugar em *Erice, Italia*, em novembro de 1999.

Esta Tese é dividida em duas partes. Na Parte I, entitulada "Convergência do Algoritmo do Volume", introduz-se o novo padrão algorítmico de extra-subgradientes para problemas de otimização não-suaves (Non Smooth Optimization, NSO), e a versão revisada do algoritmo do volume (RVA) para programação linear, junto com uma prova de sua convergência e um limite *a posteriori* para o erro de aproximação das soluções primais estimadas. Na Parte II, entitulada "Resolução do Problema de Steiner em Grafos com o Algoritmo do Volume", introduz-se dois novos algoritmos para a resolução do Problema de Steiner em Grafos (SPG), baseados no algoritmo do volume e no algoritmo do volume revisado, respectivamente, incluídas três novas heurísticas baseadas nas soluções primais estimadas por cada algoritmo; apresenta-se também uma comparação entre o algoritmo do volume e o algoritmo do volume revisado para algumas classes de problemas de Steiner.

Parte I

Convergência do Algoritmo do Volume

1 Convergência do Algoritmo do Volume

1.1 Introdução

Considere o seguinte problema de programação linear:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} c^T x \\ \quad Ax = b \qquad \text{(a)} \\ \quad Dx = e \\ \quad x \geq 0, \end{cases} \tag{1}$$

onde $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $D \in \mathbb{R}^{p \times n}$, $e \in \mathbb{R}^p$, e posto$(D) = p$.

Quando (1) é de grande porte e é difícil lidar com as restrições (1)(a), uma abordagem possível consiste em utilizar relaxação Lagrangeana e resolver o problema dual associado.

Para que esta abordagem seja eficiente, há dois pontos cruciais:

- como resolver o problema dual não-diferenciável,
- como recuperar uma solução primal.

O *Algoritmo do Volume* (VA) introduzido em [5] apareceu como uma boa resposta às perguntas anteriores:

- uma solução do dual de (1) é obtida aplicando-se o que os autores chamam de uma *extensão* do algoritmo de supergradientes [36, 64]. Tal extensão visa a manter a simplicidade dos métodos de supergradientes ao mesmo tempo que faz uso de uma estratégia baseada nos métodos de feixes (*bundle methods*), utilizada em [46, 74];
- uma aproximação para uma solução primal é simultaneamente gerada através da estimativa dos volumes associados às faces de (1)(a) que estão ativas em uma vizinhança de uma solução dual. Mais precisamente, tal aproximação é baseada no Teorema 2.2 de [5] e será vista com mais detalhes na subseção 1.3.2.

Os excelentes resultados computacionais apresentados em [5, 6, 7, 8] para problemas de *set partitioning, set covering, max-cut* e *facility location* validam a utilização do algoritmo do volume para problemas de otimização combinatória.

Na realidade, a ênfase de [5] é mais em resultados numéricos, as propriedades de convergência de VA não são analisadas. Na primeira parte desta tese enfoca-se esta questão ao introduzir uma versão *revisada* do algoritmo do volume, chamada de RVA, que difere ligeiramente de VA e para a qual prova-se convergência.

Ao longo da prova de convergência de RVA, mostra-se que VA na verdade não se encaixa na classe dos métodos de supergradientes, sendo melhor classificado como um método de extragradientes [44, 65]. Mais precisamente, cada iteração de VA gera o que se denomina um *ponto extra*, através de um método de ε-extragradientes, que será apresentado em § 1.4.1. Uma iteração é declarada *verde* em [5], quando há uma *melhora* na função dual, um tanto quanto parecido com um passo-sério nos métodos de feixes. Entretanto, esta melhora não é medida. A versão revisada proposta neste trabalho introduz uma *medida precisa* para a melhora necessária para declarar-se uma iteração verde. Tal medida é análoga ao "teste de passo-sério" do método de feixes e é a chave para a prova da convergência.

Uma interessante conseqüência da intenção inicial de provar a convergência de VA foi o desenvolvimento de um novo padrão algorítmico para problemas de otimização não-suaves (Non Smooth Optimization, NSO) no qual um número infinito de passos-sérios é gerado. Uma vez que sua principal característica consiste em atualizar os multiplicadores de Lagrange através de um deslocamento de ε-extragradiente, chama-se este método de ε-ESG.

A primeira parte desta seção é organizada da seguinte maneira: em § 1.2 revisam-se algumas noções essenciais de dualidade, relaxação Lagrangeana, convexidade e concavidade; em § 1.3 apresenta-se um panorama geral dos principais métodos de resolução para problemas de Otimização Não-Diferenciável, e introduzem-se as idéias básicas do algoritmo do volume, motivação para os trabalhos desenvolvidos nesta Tese.

Na segunda parte desta seção, § 1.4, são apresentados os algoritmos desenvolvidos neste trabalho: em § 1.4.1 introduz-se o novo padrão algorítmico de extragradientes para problemas de otimização não-suaves (Non Smooth Optimization, NSO), chamado de ε-ESG, e suas propriedades de convergência; e a subseção § 1.4.2 é devotada ao algoritmo do volume Revisado, chamado de RVA: apresentação, análise de convergência e recuperação primal; em particular, introduz-se um limite *a posteriori* para o erro de aproximação relacionando o ponto estimado primal gerado por RVA com uma solução primal.

1.2 Relaxação Lagrangeana e Análise Convexa

Nesta subseção faz-se uma revisão dos principais conceitos envolvendo relaxação Lagrangeana [10, 16, Capítulo III] e análise convexa e côncava [18, Capítulo VIII]. Em particular, em § 1.2.2, introduz-se o importante conceito de ε-supergradiente, que será utilizado ao longo desta Tese.

1.2.1 Problema Dual

Conforme dito anteriormente em § 1.1, quando o problema (1) é de grande porte e as restrições (1)(a) complicam muito a sua resolução, uma abordagem possível consiste em utilizar relaxação Lagrangeana e resolver o problema dual associado.

A principal idéia da relaxação Lagrangeana é a seguinte: em vez de trabalhar explicitamente com as restrições difíceis (1)(a), permite-se que elas sejam violadas, associando-se a elas um vetor de penalidades, ou multiplicadores de Lagrange, $\pi \in \mathbb{R}^m$, onde m é o número de linhas da matriz A de (1)(a).

Ao dualizar-se as restrições (1)(a) com os multiplicadores de Lagrange π, obtém-se o seguinte *problema Lagrangeano*:

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} \; c^T x + \pi^T (Ax - b) \\ \quad D\,x \;=\; e \\ \quad\;\; x \;\geq\; 0\,, \end{cases} \tag{2}$$

onde

$$L(x, \pi) := c^T x + \pi^T (Ax - b)\,, \tag{3}$$

é chamada de função Lagrangeana.

Seja o poliedro não-vazio definido pelas restrições "fáceis" de (1) denotado por:

$$\Psi := \{x \in \mathbb{R}^n : Dx = e\,,\; x \geq 0\}\,. \tag{4}$$

Seja $\theta(\pi)$ o custo ótimo do problema relaxado (2) para um dado π, e seja $x^\star$ uma solução (ótima) do problema (1). Dado que o poliedro Ψ inclui a região viável do problema (1) e que, para todo ponto viável x de (1), em particular para $x = x^\star$, verifica-se que:

$$\pi^T (Ax - b) \;=\; 0\,,$$

pode-se concluir que:

$$\begin{aligned}\theta(\pi) &= \min_{x\in\Psi} L(x,\pi)\\ &\leq L(x^\star,\pi)\\ &= c^T x^\star,\end{aligned}$$

Sendo assim, cada vetor de multiplicadores de Lagrange π conduz a um *limite inferior* $\theta(\pi)$ para o custo ótimo do problema (1). Logo, o interessante é encontrar o maior limite inferior possível, o que pode ser feito resolvendo-se o seguinte problema de programação linear, chamado de *problema dual*:

$$\max_{\pi\in\mathbb{R}^m} \theta(\pi)\,, \tag{5}$$

cujo objetivo é dado pela *função dual*:

$$\theta(\pi) := \min_{x\in\Psi} L(x,\pi)\,. \tag{6}$$

Sem quaisquer suposições extras, a relação de dualidade fraca

$$\min_{x\in\Psi}\max_{\pi\in\mathbb{R}^m} L(x,\pi) \geq \max_{\pi\in\mathbb{R}^m}\min_{x\in\Psi} L(x,\pi) \tag{7}$$

é verificada. O lado esquerdo de (7) é equivalente ao *problema primal* (1), e o lado direito de (7) é o problema dual (5).

Sabe-se da teoria de programação linear [17, §4, Teorema 4.4] que se um problema linear possui solução (ótima), então para os pontos primais e duais ótimos vale a igualdade em (7). Este resultado é conhecido como *dualidade forte* e neste caso diz-se que não há *salto de dualidade*. Isso signifca que resolver (1) é equivalente a resolver (5).

Na maioria dos problemas de programação inteira, como o problema que será visto em § 2, há salto de dualidade. Por isso, em geral, trabalha-se com a relaxação linear deste problema e depois tenta-se encontrar uma solução inteira usando-se algum método enumerativo, como por exemplo *branch-and-bound* [75], e/ou alguma heurística primal [19].

1.2.2 Supergradientes e ε-supergradientes

A função dual, definida em (6) como o mínimo ponto a ponto de funções afins de π, é côncava e não-diferenciável.

Para maximizar a função $\theta(\pi)$, sabe-se que qualquer método de otimização não-suave utiliza a informação dada por um "oráculo". Mais especificamente, dado qualquer π, o oráculo retorna o valor $\theta(\pi)$ e um supergradiente da função convexa $-\theta$ em π.

Uma vez que ao longo deste texto far-se-á sempre referência à função côncava θ, é conveniente introduzir-se já as noções de *supergradiente* e *ε-supergradiente*:

Definição 1.1 Seja θ uma função côncava e seja $\bar{\pi} \in \mathbb{R}^m$. O ponto $\bar{v} \in \mathbb{R}^m$ é dito um *supergradiente* de θ em $\bar{\pi}$ quando

$$\theta(\pi) \leq \theta(\bar{\pi}) + \bar{v}^T(\pi - \bar{\pi}) \text{ , para todo } \pi \in \mathbb{R}^m. \tag{8}$$

Neste caso, diz-se que $\bar{v} \in \partial\theta(\bar{\pi})$, onde o super-diferencial de θ em $\pi \in \mathbb{R}^m$ é definido como: $\partial\theta(\pi) := \{\bar{v} \in \mathbb{R}^m : \theta(\pi) \leq \theta(\bar{\pi}) + \bar{v}^T(\pi - \bar{\pi}) \text{ para todo } \pi \in \mathbb{R}^m\}$.

Dado $\varepsilon \geq 0$, o ponto $\hat{v} \in \mathbb{R}^m$ é um *ε-supergradiente* de θ em $\hat{p} \in \mathbb{R}^m$ sempre que

$$\theta(\pi) \leq \theta(\hat{p}) + \hat{v}^T(\pi - \hat{p}) + \varepsilon \text{ , para todo } \pi \in \mathbb{R}^m . \tag{9}$$

Neste caso, escreve-se $\hat{v} \in \partial\theta_\varepsilon(\hat{p})$. □

Quando particularizado para a função dual (6), vê-se que, a fim de avaliar $\theta(\bar{\pi})$ para um dado $\bar{\pi}$, o oráculo deve resolver o seguinte subproblema:

$$\theta(\bar{\pi}) = \min_{x \in \Psi}\{c^T x + \bar{\pi}^T(Ax - b)\}. \tag{10}$$

Seja $x(\pi) \in \operatorname{Argmin}\theta(\pi)$ uma solução de (10) ao trocar-se $\bar{\pi}$ por π. É fácil ver que:

$$\begin{aligned}
\theta(\pi) &= L(x(\pi), \pi) \\
&\leq L(x(\bar{\pi}), \pi) \\
&= c^T x(\bar{\pi}) + \pi^T(Ax(\bar{\pi}) - b) \\
&= c^T x(\bar{\pi}) + \pi^T(Ax(\bar{\pi}) - b) - \bar{\pi}^T(Ax(\bar{\pi}) - b) + \bar{\pi}^T(Ax(\bar{\pi}) - b) \\
&= c^T x(\bar{\pi}) + \bar{\pi}^T(Ax(\bar{\pi}) - b) + (\pi - \bar{\pi})^T(Ax(\bar{\pi}) - b) \\
&= \theta(\bar{\pi}) + (\pi - \bar{\pi})^T(Ax(\bar{\pi}) - b).
\end{aligned}$$

Daí segue que o supergradiente $\bar{v} := Ax(\bar{\pi}) - b \in \partial\theta(\bar{\pi})$ pode ser computado facilmente uma vez que o oráculo tenha resolvido (10). Claramente a dificuldade em

resolver (10) depende da natureza das restrições que definem o poliedro (4). Além disso, quanto mais fácil seja a resolução do subproblema, mais eficiente será essa abordagem dual. Este é precisamente o caso de (1), onde (4) é um poliedro convexo.

A importância do conceito de ε-super-diferencial $\partial_\varepsilon\theta(\hat{p})$ origina-se de suas boas propriedades de continuidade como uma multi-função de ε e $\hat{p}$. Resumem-se em seguida dois resultados que serão úteis ao longo do texto:

Teorema 1.1 *Considere o conjunto $\partial_\varepsilon\theta(\hat{p})$ formado por ε-supergradientes como definido em (9). Os seguintes resultados são válidos:*

(i) Fechamento: suponha que as seqüências $\{\varepsilon_t\}_t$, $\{\hat{p}_t\}_t$, $\{\hat{v}_t \in \partial_{\varepsilon_t}\theta(\hat{p}_t)\}_t$ convirjam para $\varepsilon^*, \hat{p}^*$ e $\hat{v}^*$, respectivamente. Então $\hat{v}^* \in \partial_{\varepsilon^*}\theta(\hat{p}^*)$.

(ii) dado $\hat{v} \in \partial_{\hat{\varepsilon}}\theta(\hat{p})$, para cada β positivo existe um único q_β satisfazendo as relações

$$\|q_\beta\|^2 \leq \hat{\varepsilon} \quad e \quad \hat{v} + \frac{1}{\beta} q_\beta \in \partial\theta(\hat{p} + \beta q_\beta)\,.$$

Prova: Uma prova de (i) pode ser encontrada em [37, Capítulo XI, Teorema 4.2.1]. Já (ii) é uma conseqüência do Teorema de Brønsted-Rockafellar e uma prova pode ser encontrada em [63, § 3, Teorema 3]. □

Termina-se esta seção com um resultado de ε-monotonicidade relacionando supergradientes e ε-supergradientes:

Proposição 1.1 *A relação $(\bar{v} - \hat{v})^T(\bar{\pi} - \hat{p}) \leq \hat{\varepsilon}$ verifica-se para quaisquer $\bar{v} \in \partial\theta(\bar{\pi})$ e $\hat{v} \in \partial_{\hat{\varepsilon}}\theta(\hat{p})$ dados.*

Prova: Sejam: a desigualdade do supergradiente (8) para $\bar{v}$ em $\pi = \hat{p}$:

$$\theta(\hat{p}) \leq \theta(\bar{\pi}) + \bar{v}^T(\hat{p} - \bar{\pi})\,,$$

e a desigualdade de $\hat{\varepsilon}$-supergradiente (9) para $\hat{v}$ em $\pi = \bar{\pi}$:

$$\theta(\bar{\pi}) \leq \theta(\hat{p}) + \hat{v}^T(\bar{\pi} - \hat{p}) + \hat{\varepsilon}\,.$$

O resultado é imediato. □

1.3 Resolução do Problema Dual

Já foi mencionado em § 1.2.2 que é necessário um método de otimização não-suave para resolver o problema dual (5). Para a minimização irrestrita sobre uma função convexa f, $f \equiv -\theta$ neste trabalho, muitos métodos NSO estão disponíveis: supergradientes [36, 64], planos de corte (*cutting-planes*) [41], métodos de feixes (*bundle methods*) [37], entre outros.

Todos esses métodos possuem vantagens e desvantagens, e as respectivas eficiências dependem da natureza do problema a ser resolvido. Por exemplo, os métodos de supergradientes são conhecidos por sua simplicidade, mas também pela falta de um critério de parada bem definido. Por outro lado, os métodos de feixes são robustos e precisos, embora requeiram a resolução de um problema de programação quadrática potencialmente pesado a cada iteração. Em § 1.3.1 apresenta-se um panorama geral desses métodos, destancando-se suas principais características.

Em § 1.3.2 são apresentadas as principais idéias do algoritmo do volume [5], um algoritmo que guarda a simplicidade do algoritmo de supergradientes, além de gerar boas aproximações para soluções primais. O problema deste algoritmo reside em não haver uma medida precisa para a melhora da função dual a cada iteração.

1.3.1 Panorama Geral dos Métodos de Otimização Não-Diferenciável

Nesta subseção apresenta-se um panorama geral dos métodos para a resolução de problemas de otimização não-diferenciável, visando a destacar suas principais características e comparar suas vantagens e desvantagens.

Conforme dito em § 1.2.2, para maximizar a função $\theta(\pi)$, sabe-se que qualquer método de otimização não-diferenciável utiliza a informação dada por um "oráculo". Mais especificamente, dado qualquer π, o oráculo retorna o valor $\theta(\pi)$ e um supergradiente da função côncava θ em π.

Quando particularizado para a função dual (6), a fim de avaliar $\theta(\pi)$, o oráculo deve resolver o subproblema (10). Sendo $x(\pi) \in \operatorname{Argmin} \theta(\pi)$ uma solução do subproblema, o supergradiente associado é facilmente computado: $v := Ax(\pi) - b \in \partial\theta(\pi)$.

Métodos de Supergradientes. Relembre que no caso da otimização diferenciável, a direção $\nabla\theta(\pi)$ é uma direção de subida para problemas de maximização. De fato, qualquer direção dentro do semi-espaço do gradiente (fazendo, portanto, um ângulo agudo com $\nabla\theta(\pi)$) é uma direção de subida. O método de supergradientes [36, 64] faz

uso de uma idéia análoga, utilizando porém a informação gerada pela resolução dos subproblemas: os supergradientes, que não necessariamente geram direções de subida.

A seguir o padrão algorítmico de supergradientes [18, § 6.4].

Padrão algorítmico de supergradientes

PASSO 0. *[Inicializações]*
Seja $\bar{\pi}_1 \in \mathbb{R}^m$. Inicializar $t := 1$.

PASSO 1. *[Resolução do Subproblema e Teste de Parada]*
Resolver (10) com $\bar{\pi} := \bar{\pi}_t$. Sejam: $\bar{x}_t$ um *minimizador* e $\bar{v}_t := A\bar{x}_t - b \in \partial\theta(\bar{\pi}_t)$ o *supergradiente* asssociado. Se $0 \in \partial\theta(\bar{\pi}_t)$, PARAR.

PASSO 2. *[Direção de Deslocamento]*
Calcular a *direção de deslocamento*:

$$d_t := \frac{\bar{v}_t}{\|\bar{v}_t\|}.$$

PASSO 3. *[Deslocamento de Supergradiente]*
De posse da direção d_t, encontrar um *tamanho de passo* $s_t > 0$, e efetuar o *deslocamento de supergradiente*:

$$\bar{\pi}_{t+1} := \bar{\pi}_t + s_t d_t.$$

PASSO 4. *[Loop]*
Isso completa a $t^{\underline{a}}$ iteração: fazer $t := t + 1$ e retornar ao PASSO 1. □

Os métodos de supergradientes caracterizam-se essencialmente por:

- não utilizar a informação gerada nas iterações anteriores (comportamento "Markoviano"), uma vez que cada multiplicador de Lagrange é atualizado em função de um único supergradiente (ou supergradiente); *[desvantagem]*
- não possuir um teste de parada bem definido. O teste de parada do PASSO 1 não é implementável. Em geral são usados uma tolerância máxima para o número de iterações e/ou uma tolerância mínima para o tamanho de passo s_t; *[desvantagem]*
- não utilizar nenhuma "medida de melhora", relativa à função dual, de uma iteração para outra; *[desvantagem]*
- ser muito simples, cada novo multiplicador de Lagrange é atualizado em base a um tamanho de passo na direção do supergradiente: o *deslocamento de supergradiente*

do PASSO 3. Quando se conhece um limite superior para a solução do problema, é suficiente definir o *tamanho de passo* s_t do PASSO 3 como:

$$s_t := \mu \frac{\text{UB} - \theta(\bar{\pi}_t)}{\|\bar{v}_t\|^2},$$

sendo $\mu \in (0, 2)$ um fator de correção do tamanho de passo e UB um limite superior conhecido, para obter-se convergência geométrica [18, § 6].

- apresentar um comportamento lento perto do ótimo; [*desvantagem*]

Observação 1.1 O teste de parada $0 \in \partial\theta(\bar{\pi}_t)$ não é implementável. Geralmente utiliza-se como teste de parada para o algoritmo de supergradientes um limite para o número máximo de iterações.

Métodos de Planos de Corte. Com relação aos métodos de supergradientes, os métodos de planos de corte ([41]) introduzem duas melhoras importantes: uma "memória" das iterações anteriores e um modelo de aproximação linear por partes para a função objetivo dual, o que gera um critério de parada bem definido.

Mais especificamente, de posse dos valores da função dual $\theta(\bar{\pi}_i)$ e dos supergradientes $\bar{v}_i$; $i = 1, \ldots, t$, obtidos nas iterações anteriores, constrói-se a aproximação côncava linear por partes, chamada de *modelo*:

$$\check{\theta}_t(\pi) := \min_{i=1,\ldots,t} \{\theta(\bar{\pi}_i) + \bar{v}_i^T(\pi - \bar{\pi}_i)\}. \tag{11}$$

A cada iteração do algoritmo de planos de corte, a maximização do modelo $\check{\theta}_t(\pi)$, sobre um compacto convexo C a ser determinado, gera um novo multiplicador $\bar{\pi}_{t+1}$.

A desigualdade do supergradiente (8) em conjunto com (11) implica que $\check{\theta}_t(\pi) \geq \theta(\pi)$, quaisquer que sejam $t > 0$ e $\pi \in \mathbb{R}^m$. É fácil ver também que $\check{\theta}_{t+1} \geq \check{\theta}_t$. A seguir apresenta-se o padrão algorítmico de planos de corte ([18, § 6.4]).

Padrão algorítmico de planos de corte

PASSO 0. [*Inicializações*]

Seja $\delta_{\text{min}} > 0$ uma tolerância mínima de parada, e seja $C \neq \emptyset$ um compacto contendo um ótimo de θ.

Escolher $\bar{\pi}_1 \in C$, inicializar $t := 1$, e definir $\check{\theta}_0(\pi) := +\infty$.

PASSO 1. [*Resolução do Subproblema e Teste de Parada Implementável*]

Resolver (10) com $\bar{\pi} = \bar{\pi}_t$. Sejam: $\bar{x}_t$ um *minimizador* e $\bar{v}_t := A\bar{x}_t - b \in \partial\theta(\bar{\pi}_t)$ o *supergradiente* asssociado.

Calcular o *acréscimo nominal*:

$$\delta_t := \check{\theta}_{t-1}(\pi) - \theta(\bar{\pi}_t)\,.$$

Se $\delta_t \leq \delta_{\text{min}}$, PARAR.

PASSO 2. [*Direção de Deslocamento*]

Encontrar:

$$\bar{\pi}_{t+1} \in \underset{\pi \in C}{\text{Argmax}}\ \check{\theta}_t(\pi)\,. \tag{12}$$

Definir a *direção de deslocamento*:

$$d_t := \bar{\pi}_{t+1} - \bar{\pi}_t\,.$$

PASSO 3. [*Deslocamento de Supergradiente*]

De posse da direção d_t, efetuar o *deslocamento de supergradiente*:

$$\bar{\pi}_{t+1} := \bar{\pi}_t + d_t\,.$$

PASSO 4. [*Loop*]

Isso completa a $t^{\underline{a}}$ iteração: fazer $t := t + 1$ e retornar ao PASSO 1. □

Os métodos de planos de corte caracterizam-se essencialmente por:

- introduzir uma "memória", que permite gerar uma seqüência de direções que aproveitam a informação gerada pelos supergradientes das iterações anteriores; [*vantagem*]
- ter um critério de parada bem definido. A introdução do modelo linear por partes (11) para a função dual permite quantificar (PASSO 1) um "acréscimo nominal" $\delta_t > 0$, predito por $\check{\theta}_{t-1}(\pi)$. Com isso pode-se gerar um teste de parada implementável baseado na distância entre o modelo e a função dual, a cada iteração; [*vantagem*]
- assim como o algoritmo de supergradientes apresentado anteriormente, o algoritmo de planos de corte gera a seqüência $\{\bar{\pi}_t\}_t$, que não é de subida para $\theta(\bar{\pi}_t)$. Não obstante, esta seqüência é de subida para $\check{\theta}_{t-1}(\pi)$, já que $\check{\theta}_{t-1}(\bar{\pi}_t) \geq \check{\theta}_{t-1}(\bar{\pi}_{t-1})$; [*desvantagem*]
- não ser mais assim tão simples quanto os métodos de supergradientes, uma vez que agora para calcular o próximo multiplicador de Lagrange necessita-se resolver o problema de programação linear (12); [*desvantagem*]

- oscilar perto do ótimo; [*desvantagem*]

Observação 1.2 Os métodos de planos de corte apresentam melhoras importantes em relação aos métodos de supergradientes, principalmente no tocante à introdução do modelo linear por partes para a função dual, gerando um critério de parada bem definido, que não existia anteriormente. Porém, a escolha do compacto C no PASSO 0 do algoritmo é um dos aspectos chave que determinam a instabilidade intrínseca dos algoritmos de planos de corte. Isso faz com que esses algoritmos possam apresentar performances numéricas desastrosas. Em [18, Exemplo 6.13, § 6.4] é apresentado em detalhes um exemplo proposto por Nemirovskii que comprova os efeitos desta instabilidade. Os métodos de feixes, apresentados a seguir, aparecem como uma resposta à essa questão, apresentando regularizações que estabilizam o modelo. □

Observação 1.3 Uma outra desvantagem dos métodos de planos de corte reside na acumulação indefinida de feixes no modelo. Na Observação 1.6 a seguir, ver-se-á que nos métodos de feixes há uma técnica de agregação que lhes permite conservar um número limitado de elementos, sem no entanto perder as propriedades de convergência. □

Métodos de Feixes. Conforme dito anteriormente, os métodos de supergradientes e de planos de corte não são métodos de subida para $\theta(\bar{\pi}_t)$. Os métodos de feixes [37] reúnem ao mesmo tempo as propriedades de *subida* e *estabilidade*, podendo ser considerados como variantes estabilizadas dos métodos de planos de corte. A seguir apresenta-se um dos padrões algorítmicos de feixes [18, § 7.1 e § 7.2], onde a estabilidade provém de estabelecer-se uma *região de confiança* para o modelo linear por partes.

Padrão algorítmico de feixes

PASSO 0. [*Inicializações*]

Sejam: $\delta_{\mathsf{min}} > 0$ uma tolerância-séria-mínima e $\tau_s \in (0,1)$ uma tolerância-séria. Escolher $\bar{\pi}_1$ e resolver (10) com $\bar{\pi} = \bar{\pi}_1$. Sejam: $\bar{x}_1$ um *minimizador* e $\bar{v}_1 := A\bar{x}_1 - b \in \partial\theta(\bar{\pi}_1)$ o *supergradiente* asssociado. Construir o modelo linear por partes

$$\check{\theta}_1(\pi) := \theta(\bar{\pi}_1) + \bar{v}_1^T(\pi - \bar{\pi}_1)\,.$$

Fazer $t := 1$ e inicializar $\delta_1 := \infty$.

PASSO 1. [*Teste de Parada Implementável*]

Se $\delta_t \leq \delta_{\mathsf{min}}$, PARAR.

PASSO 2. [*Candidata à Direção de Subida*]

Seja $B_t := \{\pi \in \mathbb{R}^m : \|\pi - \bar{\pi}_t\|_t^2 \leq \kappa_t\}$ uma bola de centro em $\bar{\pi}_t$ e raio κ_t, de tal forma que $\kappa_t > 0$ e $\kappa_t \to 0$ para $t \to \infty$.

Encontrar:

$$\bar{\pi}_c \in \operatorname*{Argmax}_{\pi \in B_t} \check{\theta}_t(\pi)\,. \tag{13}$$

PASSO 3. [*Resolução do Subproblema e avaliação da direção candidata*]

Resolver (10) com $\bar{\pi} = \bar{\pi}_c$. Sejam: $\bar{x}_c$ um *minimizador* e $\bar{v}_c := A\bar{x}_c - b \in \partial\theta(\bar{\pi}_c)$ o *supergradiente* asssociado.

Calcular o *acréscimo nominal* relativo ao *centro de estabilidade* $\bar{\pi}_c$:

$$\delta_t := \check{\theta}_t(\bar{\pi}_c) - \theta(\bar{\pi}_t)\,.$$

Efetuar o *teste de subida* ou *teste-sério*:

$$\theta(\bar{\pi}_c) - \theta(\bar{\pi}_t) \;\geq\; \tau_s\,\delta_t\,, \tag{14}$$

para decidir quando efetuar:

– ou um *passo-sério* (ou de subida): (14) verifica-se.
Atualizar o centro de estabilidade:

$$\bar{\pi}_{t+1} := \bar{\pi}_c\,;$$

– ou um *passo-nulo*: (14) não se verifica.
O multiplicador não se altera:

$$\bar{\pi}_{t+1} := \bar{\pi}_t\,.$$

PASSO 4. [*Melhora do Modelo*]

Incorporar $\bar{\pi}_c$ ao feixe, atualizando o modelo linear por partes:

$$\check{\theta}_{t+1}(\pi) := \min\left\{\check{\theta}_t(\pi)\,,\, \theta(\bar{\pi}_c) + \bar{v}_c^T(\pi - \bar{\pi}_c)\right\}.$$

PASSO 5. [*Loop*]

Isso completa a $t^{\underline{a}}$ iteração: fazer $t := t+1$ e retornar ao PASSO 1. ■

Os métodos de feixes caracterizam-se por:

- manter uma "memória" das iterações que geram passos-"sérios", o que permite gerar uma seqüência de direções de subida que é na verdade uma subseqüência das direções candidatas à subida; [*vantagem*]

- introduzir regularizações que *estabilizam* o modelo linear por partes para a função dual. No padrão algorítmico apresentado anteriormente tem-se uma regularização quadrática, proveniente do estabelecimento uma região de confiança para o modelo. Assim como nos métodos de planos de corte, tem-se um critério de parada bem definido; [*vantagem*]

- introduzir a noção de *centro de estabiliadde* e uma "medida de melhora" do modelo em relação às iterações anteriores, dada pela pela combinação da *tolerância-séria* τ_s com o *acréscimo nominal* δ_t. Como $\delta_t > 0$, o *teste de subida* ou *teste-sério* (14) do PASSO 3 mede se o ponto candidato $\bar{\pi}_c$ conduz a uma subida em θ:
  - quando a subida é de pelo menos uma fração τ_s do acréscimo nominal δ_t predito pelo modelo, o centro de estabilidade é deslocado para $\bar{\pi}_c$. Neste caso declara-se um *passo-sério* ou *passo de subida*;
  - quando o ponto candidato $\bar{\pi}_c$ não conduz a uma subida significativa em θ, o centro de estabilidade não se altera, i.e., o candidato não é suficientemente bom. Como nada é feito neste passo, declara-se um *passo-nulo*.

  Os *centros de estabilidade* são aqueles multiplicadores de Lagrange que verificam o teste-sério, ou seja, aqueles multiplicadores que geram uma melhora substancial na função dual; [*vantagem*]

- ser o menos simples dos métodos de otimização não-diferenciável, uma vez que para calcular o próximo multiplicador de Lagrange necessita-se resolver um problema de programação quadrática como (13); [*desvantagem*]

- não oscilar perto do ótimo; [*vantagem*]

Observação 1.4 Outro tipo de regularização é obtida quando se considera o dual do *problema de região de confiança* (13), chamado de *problema de nível*. Neste caso, objetiva-se minimizar o raio da bola em torno do centro de estabilidade onde deve-se encontrar o ponto candidato, de modo a assegurar um certo nível de subida ℓ_t (determinado a cada iteração, assim como κ_t em (13)) a ser atendido pelo modelo:

$$\bar{\pi}_c \in \underset{\pi\in\mathbb{R}^m:\check{\theta}_t(\pi)\leq\ell_t}{\operatorname{Argmin}} \frac{1}{2}\|\pi-\bar{\pi}_t\|_t^2 . \tag{15}$$

□

Observação 1.5 Uma outra variante do padrão algorítmico apresentado anteriormente consiste em agregar uma *penalização* ao modelo. A cada iteração determina-se um parâmetro μ_t e resolve-se o problema penalizado:

$$\bar{\pi}_c = \underset{\pi\in\mathbb{R}^m}{\operatorname{argmax}} \check{\theta}_t(\pi) - \frac{1}{2}\mu_t\|\pi-\bar{\pi}_t\|_t^2 . \tag{16}$$

Essa variante é conhecida como "método de feixes penalizado". □

Em [18, § 7.3, Teorema 7.7] prova-se que os métodos definidos pelas variantes (13), (15) e (16) são equivalentes. Na prática, esses métodos distingüem-se pela maneira como seus respectivos parâmetros de ajuste κ_t, ℓ_t e μ_t são atualizados. As performances numéricas variam em função destas distintas atualizações.

Observação 1.6 Os métodos de planos de corte e as variantes de feixes até agora apresentadas possuem ainda uma desvantagem: a acumulação indefinida de feixes no modelo. Em [18, § 7.3.2] pode-se encontrar uma técnica de agregação que permite aos métodos de feixes conservar um número limitado de elementos, sem perder-se as propriedades de convergência. Vale ressaltar que esta propriedade não é compartilhada pelos métodos de planos de corte. ■

Mais uma vez, todos esses métodos possuem vantagens e desvantagens, e as respectivas eficiências dependem da natureza do problema a ser resolvido. É fácil reconhecer, porém, dois extremos: de um lado os métodos de supergradientes, caracterizados por sua simplicidade e pela falta de um critério de parada bem definido, e de outro lado os métodos de feixes, caracterizados por sua robustez e precisão, requerendo, no entanto, a resolução de um problema de programação quadrática potencialmente pesado a cada iteração.

1.3.2 Algoritmo do Volume

Nesta subseção apresenta-se a motivação principal deste trabalho: o algoritmo do volume, introduzido por Barahona e Anbil em [5].

A grande novidade do algoritmo do volume reside na maneira como os valores das variáveis primais são estimados. Demonstra-se em [5, § 2, Teorema 2.2] que as variáveis primais podem ser estimadas pelo *volume* definido sob as faces das desigualdades que estão ativas em uma solução dual.

Na verdade, o algoritmo do volume mostrou-se muito bom não só por produzir boas estimativas primais, como também por gerar limites inferiores de muito boa qualidade, como será mostrado em § 2.4 para o problema de Steiner em grafos e como pode ser visto em [5, 6, 7, 8] para outras aplicações. Vale à pena ressaltar ainda que este algoritmo guarda a simplicidade do algoritmo de supergradientes, como será visto em detalhes mais à frente.

Suponha que o politopo Ψ definido na página 6 possua q pontos extremos $g_i \in \mathbb{R}^n$, $1 \leq i \leq q$. Sendo assim, $x \in \Psi$ pode ser escrito como combinação convexa desses pontos extremos:

$$\Psi := \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^{q} \lambda_i \, g_i \, , \; \sum_{i=1}^{q} \lambda_i = 1 \, , \; \lambda \geq 0 \right\} . \tag{17}$$

O problema de programação linear (1) é então equivalente ao seguinte problema linear:

$$\begin{cases} \min\limits_{\lambda \in \mathbb{R}^q} \; \sum\limits_{i=1}^{q} (c \, g_i) \, \lambda_i \\ \quad \sum\limits_{i=1}^{q} (A \, g_i - b) \, \lambda_i \; = \; 0 \\ \qquad\qquad \sum\limits_{i=1}^{q} \lambda_i \; = \; 1 \\ \qquad\qquad\qquad \lambda \; \geq \; 0 \, , \end{cases} \tag{18}$$

já que: $\sum\limits_{i=1}^{q} (A \, g_i) \lambda_i - b = \sum\limits_{i=1}^{q} (A \, g_i) \lambda_i - b \sum\limits_{i=1}^{q} \lambda_i$, desde que $\sum\limits_{i=1}^{q} \lambda_i = 1$.

Quando o número de pontos extremos é muito grande, torna-se inviável resolver o problema (18) diretamente, sendo necessário decompô-lo. Uma das técnicas utilizadas é a *Decomposição de Dantzig-Wolfe* [26], que consiste em trabalhar com subconjuntos das colunas da matriz A, que vão sendo gerados conforme necessário. Cada nova coluna é obtida resolvendo-se o problema de programação linear (2) com $\pi = \bar{\pi}$, onde $\bar{\pi} \in \mathbb{R}^m$ é o vetor das variáveis duais associadas à base corrente de (18). O problema (18) é chamado de *problema mestre* e o problema (2) é chamado de *problema escravo* ou *subproblema.*

Quando o número de colunas a gerar é *muito* grande, este procedimento pode ficar lento demais. Uma maneira de acelerar este método consiste em combiná-lo com o método de supergradientes, resolvendo o problema dual associado ao problema mestre (ver por exemplo [9]). O algoritmo do volume pode ser visto como uma maneira rápida de aproximar a decomposição de Dantzig-Wolfe. Mais explicitamente, resolve-se o problema (2) como na decomposição de Dantzig-Wolfe mas os valores de λ_i são estimados em vez de determinados pela resolução de (18).

O problema dual associado a (18) é dado por:

$$\begin{cases} \max\limits_{z\in\mathbb{R},\pi\in\mathbb{R}^m} z \\ \quad z + \pi^T (A\, g_i - b) \leq c^T g_i \\ \qquad\qquad\quad z \text{ irrestrita} \\ \qquad\qquad\quad \pi \text{ irrestrita}. \end{cases} \tag{19}$$

Antes de apresentar o algoritmo do Volume, é necessário introduzir a relação entre as variáveis primais de (18) e os volumes definidos pelas faces em uma solução de (19), dada pelo *Teorema do Volume*:

Teorema 1.2 [5, § 2, Teorema 2.2] *Seja* $(\pi^\star, z^\star)$ *uma solução de* (19) *e suponha que as restrições* $1, \ldots, m'$, $m' \leq m$, *estão ativas neste ponto. Seja* $\bar{z} < z^\star$ *e assuma que*

$$\begin{aligned} z + \pi^T (A\, g_i - b) &\leq c^T g_i, \text{ para } 1 \leq i \leq m' \\ z &\geq \bar{z} \end{aligned}$$

define um poliedro limitado. Para $1 \leq i \leq m'$, *seja* γ_i *o volume formado entre a face* $z := c^T g_i - \pi^T (A\, g_i - b)$ *e o hiperplano definido por* $z := \bar{z}$ *(ver Figura 1 na página seguinte). Então uma solução primal* λ_i *de* (18) *é dada por:*

$$\lambda_i = \frac{\gamma_i}{\sum_{j=1}^{m'} \gamma_j}. \qquad \square$$

A seguir a Figura 1 que ilustra o volume γ_i formado entre a face $z := c^T g_i - \pi^T (A\, g_i - b)$ e o hiperplano $z := \bar{z} - \epsilon$, em uma vizinhança de uma solução dual $(\pi^\star, z^\star)$.

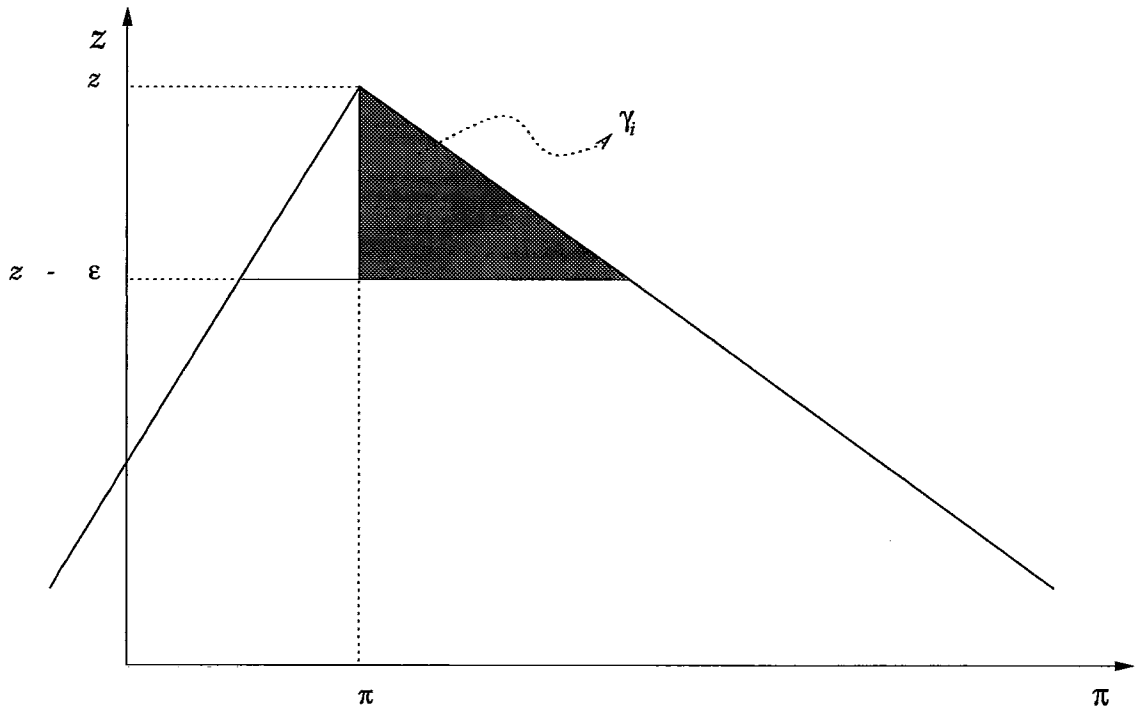

Figura 1: Volume γ_i formado entre a face $z := c^T g_i - \pi^T (A g_i - b)$ e o hiperplano $z := \bar{z} - \epsilon$ em uma vizinhança de uma solução dual $(\pi^\star, z^\star)$.

Interpretação Geométrica. Como o somatório das variáveis primais na decomposição de Dantzig-Wolfe (18) é igual a uma unidade, cada valor λ_i pode ser interpretado como a probabilidade de que, em uma vizinhança de uma solução (ótima) dual, o subproblema (2) gere o ponto extremo g_i, o que pode ser interpretado também como a probabilidade de gerar a face i. Pelo Teorema 1.2, cada λ_i é proporcional ao volume γ_i formado entre a face i e o hiperplano $z := \bar{z} - \epsilon$, para algum valor pequeno de ϵ. A região sombreada da Figura 1, acima, ilustra este volume. ■

O algoritmo do volume consiste essencialmente do algoritmo de supergradientes modificado de modo a produzir soluções primais e com critérios de parada que avaliam: salto de dualidade, inviabilidade primal e otimalidade.

A idéia principal vem de aplicar o Teorema 1.2 ao problema dual (19). Do fato de cada variável primal λ_i poder ser vista como a probabilidade de o subproblema gerar a face i na vizinhança de uma solução dual, segue que uma boa estimativa para as variáveis primais $\bar{x}$ de (2) consiste em uma combinação convexa especial dos valores das variáveis geradas anteriormente, de tal maneira que a probabilidade de a face i ser gerada em cada subproblema vá sendo acumulada nos coeficientes daquela combinação convexa. Isso será analisado com mais detalhe na primeira observação que segue à descrição do algoritmo do volume.

A seguir uma descrição detalhada do algoritmo do volume.

Algoritmo do Volume (VA)

PASSO 0. [*Inicializações*]

Sejam:

- $\bar{\pi}_0 \in \mathbb{R}^m$ um vetor de multiplicadores de Lagrange,
- z_{LB} um limite inferior para (19),
- $0 < \alpha < 1$ um escalar,
- $0 < \xi_{pd}$ uma tolerância de parada primal-dual,
- $0 < \xi_v$ uma tolerância de parada de viabilidade,
- $maxiters$ um limite para o número máximo de iterações,
- $maxtime$ um limite para o tempo máximo de CPU.

Resolver (2) com $\bar{\pi} := \bar{\pi}_0$. Sejam: $\bar{x}_0$ um minimizador, $\bar{v}_0 := A\bar{x}_0 - b \in \partial\theta(\bar{\pi}_0)$ o supergradiente asssociado e $\bar{z}_0 := \theta(\bar{\pi}_0, \bar{x}_0)$ o valor da função objetivo.

Inicializar: $z_{\text{LB}} := \bar{z}_0$, $\hat{x}_0 := \bar{x}_0$, $\hat{v}_0 := \bar{v}_0$, $\hat{\pi}_0 := \bar{\pi}_0$, $k := 0$ e $t := 0$.

PASSO 1A. [*Deslocamento de "Supergradiente"*]

Efetuar o *deslocamento de "supergradiente"*:

$$\bar{\pi}_{t+1} := \hat{\pi}_k + s_t \hat{v}_t \,,$$

onde o tamanho de passo s_t é dado por:

$$s_t := \mu \frac{\text{UB} - \theta(\bar{\pi}_t)}{\|\hat{v}_t\|^2} \,, \tag{20}$$

sendo $\mu \in (0, 2)$ um fator de correção do tamanho de passo, e UB um limite superior para (19).

PASSO 1B. [*Resolução do Subproblema*]

Resolver (2) com $\bar{\pi} := \bar{\pi}_{t+1}$. Sejam: $\bar{x}_{t+1}$ um minimizador, $\bar{v}_{t+1} := A\bar{x}_{t+1} - b \in \partial\theta(\bar{\pi}_{t+1})$ o supergradiente asssociado, e $\bar{z}_{t+1}(\bar{\pi}_{t+1}, \bar{x}_{t+1})$ o valor da função objetivo.

PASSO 2A. [*Atualização Primal*]

Computar a *aproximação primal*:

$$\hat{x}_{t+1} := \alpha\bar{x}_{t+1} \; + \; (1 - \alpha)\hat{x}_t \,. \tag{21}$$

PASSO 2B. [*Direção de Deslocamento*]

Computar a *direção de deslocamento*:

$$\hat{v}_{t+1} := A\hat{x}_{t+1} - b \,.$$

PASSO 3. [***Teste Dual***]

Efetuar o *Teste dual*:

$$\bar{z}_{t+1} > z_{\text{LB}}\,, \tag{22}$$

e:

- se (22) não se verifica, declarar uma *iteração-vermelha*;
- se (22) verifica-se, calcular $d := \hat{v}_{t+1}^T\, \bar{v}_{t+1}$, e:
  - se $d < 0$, declarar uma *iteração-amarela*;
  - se $d \geq 0$, declarar uma *iteração-verde* e atualizar:
    - o limite inferior: $z_{\text{LB}} := \bar{z}_{t+1}$;
    - k := k + 1;
    - o multiplicador: $\hat{\pi}_k := \bar{\pi}_{t+1}$.

PASSO 4. [***Testes de Parada***]

Efetuar os testes de parada:

- *Teste de parada primal-dual*: se

$$\frac{|c^T \hat{x}_{t+1} - z_{\text{LB}}|}{|z_{\text{LB}}|} < \xi_{pd}\,,$$

  PARAR;
- *Teste de parada de viabilidade*: se

$$\frac{\|\hat{v}_{t+1}\|}{m} < \xi_v\,,$$

  onde m denota o número de linhas da matriz A, PARAR ;
- *Teste de parada de otimalidade*: se

$$\text{UB} - z_{\text{LB}} < 1\,,$$

  PARAR;
- *Teste de parada de número máximo de iterações*: se

$$t > maxiters\,,$$

  PARAR;
- *Teste de parada de tempo máximo de CPU*: se

$$CPU_{time} > maxtime\,,$$

  PARAR.

PASSO 5. [***Loop***]

Fazer $t := t + 1$ e retornar ao PASSO 1A. □

Abaixo seguem algumas observações importantes e alguns detalhes de implementação:

- repare que em [***Atualização Primal***] tem-se que: $\hat{x}_{t+1} := \alpha\,\bar{x}_{t+1} + (1-\alpha)\,\hat{x}_t = \alpha\,\bar{x}_{t+1} + (1-\alpha)\,\alpha\,\bar{x}_t + \cdots + (1-\alpha)^{t+1}\,\bar{x}_0$. Os coeficientes $\alpha, (1-\alpha)\,\alpha, \ldots, (1-\alpha)^{t+1}$ são usados como uma aproximação $\bar{\lambda}$ para uma solução (ótima) λ do problema (18) na Decomposição de Dantzig-Wolfe. Durante o processo, toda vez que o subproblema (2) produz uma face i, o valor $\bar{\lambda}_i$ é atualizado como $\bar{\lambda}_i := \alpha + (1-\alpha)\bar{\lambda}_i$ e, para $j \neq i$, a atualização é dada por $\bar{\lambda}_j := (1-\alpha)\bar{\lambda}_j$. Sendo assim, se uma face i aparece apenas no início do procedimento, seu "peso" $\bar{\lambda}_i$ decresce exponencialmente. Isso é compatível com a idéia que está imbutida na componente primal de VA, que reside em tentar estimar, para cada face que esteja ativa em uma vizinhança de uma solução (ótima) de (19), a razão entre o volume formado sob esta face e a soma total dos volumes formados sob todas as faces ativas naquela solução dual (ver Teorema 1.2);

- O escalar α fica fixo no valor dado em [***Inicializações***] por um certo número de iterações e depois vai decrescendo de acordo com uma idéia utilizada originalmente nos métodos de supergradientes Conjugados [46, 74]: sejam $\alpha_{\max}$ um limite superior para α, e α_{opt} dado por:

$$\alpha_{\text{opt}} := \operatorname*{Argmin}_{\xi \in \mathbb{R}} \| \xi\,\hat{v}_{t+1} + (1-\xi)\,\bar{v}_{t+1} \|^2 .$$

  Se $\alpha_{\text{opt}} < 0$ define-se $\alpha := \frac{\alpha_{\max}}{10}$, caso contrário define-se $\alpha := \min\{\alpha_{\text{opt}}, \alpha_{\max}\}$. Isso é feito para aumentar a precisão do cálculo da aproximação primal. O parâmetro $\alpha_{\max}$ é multiplicado por um parâmetro $0 < \alpha_{\text{factor}} < 1$ a cada α_{int} iterações. Esses parâmetros dependem de cada tipo de problema;

- em [***Direção de Deslocamento***] estas estimativas para os valores primais embutidas na combinação convexa $\hat{x}_{t+1}$ são utilizadas para definir a direção de deslocamento $\hat{v}_{t+1} := A\hat{x}_{t+1} - b$, enquanto num algoritmo típico de supergradientes (página 11) essa direção é dada por apenas uma face ativa. Por isso as aspas em [***Deslocamento de "Supergradiente"***], porque na verdade, além de mostrar-se em § 1.4.2.2 que na verdade $\hat{v}_{t+1}$ é um extragradiente e não um supergradiente, não se está trabalhando com *um* "supergradiente" e sim com uma combinação convexa de "supergradientes". Outra diferença em relação ao algoritmo de supergradientes é que o vetor dual $\bar{\pi}_{t+1}$ não é atualizado em função de qualquer multiplicador $\bar{\pi}_t$, e sim em função de $\hat{\pi}_k$, que só é atualizado em [*Teste-Dual*]. Esse multiplicador faz o mesmo papel do *centro de estabilidade* do algoritmo de feixes (página 15);

- em [***Teste Dual***], definem-se os valores de atualização do fator de correção μ do tamanho de passo (20), de acordo com o tipo de iteração declarada:

  - $\mu := \mu_{\text{R}}\,\mu$, a cada R *iterações-vermelhas*;
  - $\mu := \mu_{\text{Y}}\,\mu$, a cada Y *iterações-amarelas*;

– $\mu := \mu_{\text{G}}\,\mu$, a cada G *iterações-verdes*;

onde os fatores $\mu_{\text{R}}, \mu_{\text{Y}}, \mu_{\text{G}}$, R, Y e G dependem de cada problema a que o algoritmo é aplicado, tendo-se, em geral: $0 < \mu_{\text{R}} < 1$, $1 < \mu_{\text{Y}} < 2$, $\mu_{\text{G}} \geq 2$. Em geral tem-se vários passos vermelhos antes de ter-se um passo amarelo ou verde dado por uma melhora dual;

- finalmente, em [**Testes de Parada**] tem-se o *Teste de parada primal-dual*, baseado em uma tolerância para a diferença entre o melhor limite inferior encontrado e o valor objetivo aproximado primal que está sendo calculado; o *Teste de parada de viabilidade*, baseado em uma tolerância para o quanto as restrições $Ax = b$ estão sendo violadas; o *Teste de parada de otimalidade*, que é relativo ao salto de dualidade e é válido apenas quando o problema que na verdade objetiva-se resolver é um problema inteiro com coeficientes inteiros; o *Teste de parada de número máximo de iterações*, baseado em um limite para o máximo número de iterações desejado; e, finalmente, o *Teste de parada de tempo máximo de CPU*, baseado em uma tolerância para o tempo máximo de CPU.

Conforme dito em §1.1, a ênfase de [5] é mais em resultados numéricos. As propriedades de convergência de VA não são analisadas, o que motivou todo o trabalho que será apresentado nas seguintes subseções desta primeira parte da Tese.

1.4 Novos Algoritmos para a Resolução do Problema Dual

Nesta subseção são apresentados os algoritmos desenvolvidos neste trabalho.

Em § 1.4.1 introduz-se o padrão algorítmico ε-ESG para problemas gerais de otimização não-suaves (Non Smooth Optimization, NSO) e suas propriedades de convergência são estudadas.

Em § 1.4.2 introduz-se uma versão revisada do algoritmo do volume, que denomina-se RVA. Em § 1.4.2.1 o algoritmo do volume revisado (RVA) é apresentado, em § 1.4.2.2 sua convergência é analisada, e em § 1.4.2.3 estuda-se a recuperação primal, em particular introduz-se um limite *a posteriori* para o erro de aproximação relacionando o ponto estimado primal gerado por RVA com uma solução primal.

1.4.1 Padrão Algorítmico de extragradientes

O padrão algorítmico de tipo *extragradientes* (ε-ESG) introduzido neste trabalho visa a combinar as melhores características dos métodos de supergradientes e feixes: simplicidade, robustez e precisão.

Definição 1.2 Define-se um *extragradiente* $\hat{v}$ como sendo um ε-supergradiente $\hat{v} \in \partial_{\hat{\varepsilon}}\theta(\hat{p})$, onde o *ponto extra* $\hat{p} \in \mathbb{R}^m$ é uma combinação convexa de multiplicadores de Lagrange $\bar{\pi} \in \mathbb{R}^m$.

Uma vez que o algoritmo revisado do volume (RVA) será classificado como uma instância particular deste padrão algorítmico, a seguir apresenta-se ε-ESG para a resolução de um problema côncavo de maximização geral, como em (5).

Padrão algorítmico ε-ESG

PASSO 0. Sejam: $\bar{\pi}_1 \in \mathbb{R}^m$ o multiplicador inicial, $0 < \alpha < 1$ um escalar, e $0 < \tau_s < 1$ uma *tolerância séria*. Inicializar: $k := t := 1$, $T_s := \emptyset$ e $\hat{\pi}_k := \bar{\pi}_1$ e $\hat{p}_{t-1} := \bar{\pi}_1$.

PASSO 1. Para $\bar{\pi}_t$ e $\hat{p}_{t-1}$ dados, definir o *ponto extra*

$$\hat{p}_t := \alpha\bar{\pi}_t + (1-\alpha)\hat{p}_{t-1}\,. \tag{23}$$

Um *extragradiente* $\hat{v}_t \in \partial_{\hat{\varepsilon}_t}\theta(\hat{p}_t)$ é assumido disponível.

PASSO 2. Em posse do *centro de estabilidade* $\hat{\pi}_k$, efetuar o *deslocamento de extragradiente*:

$$\bar{\pi}_{t+1} := \hat{\pi}_k + s_t\hat{v}_t\,, \tag{24}$$

para algum tamanho de passo positivo s_t; e computar a *medida de melhora*:

$$\delta_t := \hat{v}_t^T(\bar{\pi}_{t+1} - \hat{\pi}_k) + \hat{\varepsilon}_t\,. \tag{25}$$

PASSO 3. Efetuar o *teste-sério*:

$$\theta(\bar{\pi}_{t+1}) \geq \theta(\hat{\pi}_k) + \tau_s\,\delta_t\,, \tag{26}$$

e decidir quando efetuar:

- ou um *passo-nulo*: (26) não se verifica, não fazer nada,
- ou um *passo-sério*: (26) verifica-se. Atualizar o centro de estabilidade: $\hat{\pi}_{k+1} := \bar{\pi}_{t+1}$, $T_s := T_s \cup \{t\}$; e aumentar k de uma unidade.

Isso completa a $t^{\underline{a}}$ iteração: fazer $t := t+1$ e retornar ao PASSO 1 até que algum critério de parada seja satisfeito. □

Abaixo seguem algumas observações importantes:

- primeiro, quando comparada a uma iteração típica do algoritmo de supergradientes, a atualização feita em (24) não faz uso de um supergradiente em $\hat{\pi}_k$, mas sim de um ε-supergradiente em um ponto extra, a saber $\hat{p}_t$ (23). Nesse sentido ε-ESG classifica-se como um algoritmo de extragradiente [44];
- segundo, quando comparada a uma iteração típica de algoritmos de feixes, a distinção crucial entre passos-sérios e nulos é mantida, via (26). O cálculo de $\hat{v}_t$ no PASSO 1 ficará mais claro en § 1.4.2, onde ver-se-á que está baseado em uma combinação convexa simples que utiliza o escalar α de (23).

A seguir é apresentado um lema simples que é essencial para a prova de convergência de ε-ESG e que faz uso da hipótese abaixo.

Hipótese 1.1 *Assuma que θ é uma função côncava (semi-contínua superiormente) sobre $\mathbb{R}^m$, com conjunto maximizador não-vazio P^*.*

Lema 1.1 *Suponha que θ satisfaz a Hipótese 1.1. Suponha ainda que ε-ESG é implementado de tal maneira que para todo $t \in T_s$ verifica-se:*

$$0 < s_{min} \leq s_t \leq s_{max} < \infty\,. \tag{27}$$

Então $\sum_{t\in T_s} \left(\|\hat{v}_t\|^2 + \hat{\varepsilon}_t\right) < +\infty$.

Prova: Seja $t \in T_s$ dado. Ao combinar (27) e (24) obtém-se:

$$s_{\min}\|\hat{v}_t\|^2 \leq s_t\|\hat{v}_t\|^2 = \hat{v}_t^T\left(\hat{\pi}_{t+1} - \hat{\pi}_t\right),$$

o que, junto com (26), gera:

$$s_{\min}\|\hat{v}_t\|^2 \leq \frac{1}{\tau_s}\left[\theta(\hat{\pi}_{t+1}) - \theta(\hat{\pi}_t)\right] - \hat{\varepsilon}_t\,.$$

Dado $K \geq 2$ indexando os centros de estabilidade, seja T_s^K o conjunto de todos os índices t incorporados a T_s antes da geração de $\hat{\pi}_K$. Fazendo-se o somatório sobre $k < K$ obtém-se:

$$\begin{aligned}
\tau_s \sum_{t\in T_s^K}\left(s_{\min}\|\hat{v}_t\|^2 + \hat{\varepsilon}_t\right) &\leq \sum_{k=1}^{K-1}\Big(\theta(\hat{\pi}_{k+1}) - \theta(\hat{\pi}_k)\Big) \\
&= \theta(\hat{\pi}_K) - \theta(\hat{\pi}_1) \\
&\leq \max_{p\in\mathbb{R}^m}\theta(p) - \theta(\hat{\pi}_1)\,.
\end{aligned}$$

Ao juntar-se essa igualdade com as hipóteses anteriores, a prova do Lema é imediata. □

O fechamento de $\partial_{\hat{\varepsilon}}\theta(\hat{p})$ como uma (multi)função de $\hat{\varepsilon}$ e $\hat{p}$ implica o seguinte corolário:

Corolário 1.1 *Suponha válidas as hipóteses do Lema 1.1. Assuma ainda que a seqüência* $\{\hat{p}_t\}_{t\in T_s}$ *é infinita e limitada. Então* $\{\hat{p}_t\}_{t\in T_s}$ *possui um ponto de acumulação* $\hat{p}^\star$ *que resolve* (5)*, i.e.,* $\hat{p}^\star \in P^\star$.

Prova: A partir do Lema 1.1 vê-se que ambos $\hat{\varepsilon}_t$ e $\hat{v}_t$ tendem a 0 para $t \in T_s$. Em particular, para alguma subseqüência de índices $\tilde{T}_s \subset T_s$ verifica-se que:

$$(\hat{p}_t, \hat{\varepsilon}_t, \hat{v}_t) \longrightarrow (\hat{p}^\star, 0, 0)\,.$$

Relembrando o Teorema 1.1(i), a prova está terminada, porque do PASSO 1 em ε-ESG tem-se que: $\hat{v}_t \in \partial_{\hat{\varepsilon}_t}\theta(\hat{p}_t)$. □

Para que o padrão algoritmico ε-ESG seja convergente, o resultado anterior requer que a seqüência $\{\hat{p}_t\}_{t\in T_s}$ seja infinita e limitada.

Número finito de passos-sérios. Geralmente a análise de convergência dos algoritmos de feixes é dividida em duas partes, dependendo de se a cardinalidade de T_s é infinita ou não:

- no último caso, existe um último passo-sério $\hat{\pi}_{k_{\mathtt{last}}}$ gerado, seguido por um número infinito de passos-nulos. Esses passos-nulos tendem ao ponto proximal de $-\theta$ em $\hat{\pi}_{k_{\mathtt{last}}}$, o qual é mostrado ser o próprio $\hat{\pi}_{k_{\mathtt{last}}}$. Isso prova que $\hat{\pi}_{k_{\mathtt{last}}}$ é uma solução de (5);

- no contexto de ε-ESG,o raciocínio acima não se aplica diretamente porque toda a análise de convergência é feita sobre a seqüência $\{\hat{p}_t\}_{t\in T_s}$ e não sobre $\{\hat{\pi}_k\}_k$. Isto ocorre porque ε-ESG foi definido de maneira a ser praticamente idêntico (ou o mais semelhante possível) ao algoritmo do volume, o qual foi construído com o intuito de *reduzir* avaliações de funções, i.e., resoluções de subproblemas (10).

É interessante ressaltar que com a adição de um cálculo, a saber $\theta(\hat{p}_t)$, e algumas pequenas modificações, ε-ESG torna-se um algoritmo de feixes, cujas propriedades de convergência são bem estudadas em [24]. Esta interpretação baseia-se em uma expressão explícita do extragradiente $\hat{v}_t$, que é obtida quando se especializa ε-ESG a RVA. Voltar-se-á a esta questão na Observação 1.8, após ter-se apresentado a versão revisada do Algoritmo do Volume.

Fechamento. Para provar que a seqüência $\{\hat{p}_t\}_{t\in T_s}$ é limitada para uma função dual geral, é necessária uma hipótese adicional sobre θ:

Hipótese 1.2 Seja θ uma função côncava sobre $\mathbb{R}^m$, com conjunto maximizador não-vazio P^*. Seja $\text{proj}^*(\cdot)$ a aplicação de projeção sobre P^*. Então θ satisfaz a condição de *crescimento inverso* com constante $L > 0$ se existe $\rho > 0$ tal que:

$$\|w\| \leq \rho \Longrightarrow \|q - \text{proj}^*(q)\| \leq L\|w\| \text{ para todo } q \text{ such that } w \in \partial\theta(q)\,. \quad \square$$

Quando particularizada à função convexa $f \equiv -\theta$, a Hipótese 1.2 é equivalente à conjugada f^* possuindo um subdiferencial que é localmente Lipschitziano superiormente na origem [62, 63].

O resultado seguinte segue a técnica de prova de [63, § 3, Teorema 4] e assegura que a seqüência $\{\hat{p}_t\}_{t\in T_s}$ é limitada.

Proposição 1.2 *Suponha que θ satisfaz as Hipóteses 1.1 e 1.2. Suponha ainda que P^* é limitado e assuma que ε-ESG é implementado de tal maneira que para todo $t \in T_s$ (27) verifica-se. Então a seqüência $\{\hat{p}_t\}_{t\in T_s}$ é limitada.*

Prova: Para $t \in T_s$, ao aplicar o Teorema 1.1(ii) para $\hat{v}_t \in \partial_{\hat{\varepsilon}_t}\theta(\hat{p}_t)$ e $\beta := L$ conclui-se que existe un único $q_t := q_L$ satisfazendo as relações:

$$\|q_t\| \leq \hat{\varepsilon}_t^{1/2} \quad \text{e} \quad \hat{v}_t + \frac{1}{L}q_t \in \partial\theta(\hat{p}_t + Lq_t)\,. \tag{28}$$

As hipóteses implicam, via Lema 1.1, que ambos $\hat{\varepsilon}_t$ e $\hat{v}_t$ tendem a 0 para $t \in T_s$. Portanto, existe algum $\bar{t} \in T_s$ tal que, para todo $T_s \ni t > \bar{t}$, tem-se que $\|\hat{v}_t + \frac{1}{L}q_t\| \leq \rho$, com ρ dado pela Hipótese 1.2.

Logo, a condição de crescimento inverso escrita para $w = \hat{v}_t + \frac{1}{L}q_t$ e $q = \hat{p}_t + Lq_t$ gera a desigualdade:

$$\|\hat{p}_t + Lq_t - \text{proj}^*(\hat{p}_t + Lq_t)\| \leq L\|\hat{v}_t + \frac{1}{L}q_t\| \quad \text{para todo } T_s \ni t > \bar{t}\,. \tag{29}$$

Ao todo obtém-se o limite:

$$\begin{aligned}
\|\hat{p}_t - \text{proj}^*(\hat{p}_t)\| &\leq \|\hat{p}_t - \text{proj}^*(\hat{p}_t + Lq_t)\| && \text{[def. de proj}^*\text{]} \\
&\leq \|\hat{p}_t + Lq_t - \text{proj}^*(\hat{p}_t + Lq_t)\| + L\|q_t\| \\
&\leq L\|\hat{v}_t + \tfrac{1}{L}q_t\| + L\|q_t\| && \text{[por (29)]} \\
&\leq L\|\hat{v}_t\| + (1+L)\hat{\varepsilon}_t^{1/2}\,. && \text{[por (28)]}
\end{aligned}$$

Relembrando-se que P^* é limitado, a conclusão é direta. $\square$

Observação 1.7 A hipótese 1.2 pode ser julgada muito forte para provar convergência global, porque está atrelada a $-\theta$ ser fortemente convexa perto de um ótimo. Entretanto, é apenas uma condição suficiente para provar que a seqüência $\{\hat{p}_t\}_{t\in T_s}$ é limitada. Quando particularizada a (1) e seu dual, a seqüência $\{\hat{p}_t\}_{t\in T_s}$ será limitada sempre que o conjunto viável Ψ de (4) for limitado. Uma condição para que Ψ seja um politopo (i.e., um poliedro limitado) é que o sistema

$$\begin{cases} Dy &= 0 \\ y &\geq 0 \end{cases}$$

possua uma única solução, a saber $y = 0$; veja por exemplo [55, § 3.7]. □

1.4.2 Versão Revisada do Algoritmo do Volume

Antes de introduzir qualquer modificação ao algoritmo do volume (VA), vale a pena relembrar as bases da metodologia do Volume, relatadas em § 1.3. Grosseiramente falando, para produzir uma solução dual, VA gera pontos candidatos $\bar{\pi}_{t+1}$ através da resolução de um subproblema como em (10), onde $\bar{\pi}_{t+1}$ depende de:

- um centro de estabilidade $\hat{\pi}_k$,
- uma combinação convexa de supergradientes $\{\hat{v}_l\}_{l\leq t}$, e
- um tamanho de passo s_t.

Vale a pena relembrar que os centros de estabilidade são aqueles multiplicadores de Lagrange que geram uma melhora "suficientemente boa" na função dual.

Do ponto de vista da resolução dual, VA é uma instância do padrão algoritmico ε-ESG, onde o teste-sério (26) é definido simplesmente como

$$\theta(\bar{\pi}_{t+1}) > \theta(\hat{\pi}_k)\,,$$

i.e., *sem* utilizar a medida de melhora δ_t definida em (25). Na versão revisada abaixo incorpora-se esta medida ao teste-sério.

Em § 1.4.2.1 apresenta-se uma descrição detalhada do algoritmo do volume Revisado (RVA), em § 1.4.2.2 prova-se sua convergência e mostra-se como transformá-lo em um algoritmo de feixes, e em § 1.4.2.3 apresenta-se uma estimativa para a solução primal aproximada gerada.

1.4.2.1 Algoritmo do Volume Revisado

A seguir uma descrição detalhada do algoritmo do volume revisado.

Algoritmo do Volume Revisado (RVA)

PASSO 0. [*Inicializações*]

Sejam:

- $\bar{\pi}_0 \in \mathbb{R}^m$ um vetor de multiplicadores de Lagrange,
- z_{LB} um limite inferior para (19),
- $0 < \alpha < 1$ um escalar,
- $0 < \tau_s < 1$ uma *tolerância séria*,
- $0 < \delta_{\min}$ uma tolerância de parada séria,
- $0 < \xi_{pd}$ uma tolerância de parada primal-dual,
- $0 < \xi_v$ uma tolerância de parada de viabilidade,
- $maxiters$ um limite para o número máximo de iterações,
- $maxtime$ um limite para o tempo máximo de CPU.

Resolver (2) com $\bar{\pi} := \bar{\pi}_0$. Sejam: $\bar{x}_0$ um minimizador, $\bar{v}_0 := A\bar{x}_0 - b \in \partial\theta(\bar{\pi}_0)$ o supergradiente asssociado, e $\bar{z}_0 := \theta(\bar{\pi}_0, \bar{x}_0)$ o valor da função objetivo.

Inicializar: $z_{\text{LB}} := \bar{z}_0$, $\hat{x}_0 := \bar{x}_0$, $\hat{v}_0 := \bar{v}_0$, $\hat{\pi}_0 := \bar{\pi}_0$, $\hat{p}_0 := \bar{\pi}_0$, $\hat{\varepsilon}_0 := 0$, $\delta_0 := \infty$, $T_s := \emptyset$, $k := 0$ e $t := 0$.

PASSO 1A. [*Deslocamento de Extragradiente*]

Efetuar o *deslocamento de extragradiente* como em (24):

$$\bar{\pi}_{t+1} := \hat{\pi}_k + s_t \hat{v}_t\,, \tag{30}$$

onde o tamanho de passo s_t é dado por (20):

$$s_t := \mu \frac{\text{UB} - \theta(\bar{\pi}_t)}{\|\hat{v}_t\|^2}\,,$$

sendo $\mu \in (0, 2)$ um fator de correção do tamanho de passo, e UB um limite superior para (19).

PASSO 1B. [*Resolução do Subproblema*]

Resolver (2) com $\bar{\pi} := \bar{\pi}_{t+1}$. Sejam: $\bar{x}_{t+1}$ um minimizador, $\bar{v}_{t+1} := A\bar{x}_{t+1} - b \in \partial\theta(\bar{\pi}_{t+1})$ o supergradiente asssociado, e $\bar{z}_{t+1}(\bar{\pi}_{t+1}, \bar{x}_{t+1})$ o valor da função objetivo.

PASSO 2A. [*Atualização Primal*]

Computar a *aproximação primal* (21):

$$\hat{x}_{t+1} := \alpha\bar{x}_{t+1} + (1 - \alpha)\hat{x}_t\,. \tag{31}$$

PASSO 2B. [*Atualização Dual*]

Tendo $\bar{\pi}_{t+1}$ e $\hat{p}_t$, definir o *ponto extra* $\hat{p}_{t+1}$ como em (23):

$$\hat{p}_{t+1} := \alpha\bar{\pi}_{t+1} + (1-\alpha)\hat{p}_t\,. \tag{32}$$

PASSO 2C. [*Direção de Deslocamento*]

Computar a *direção de deslocamento*:

$$\hat{v}_{t+1} := A\hat{x}_{t+1} - b\,.$$

O Teorema 1.2, provado a seguir, mostrará que $\hat{v}_{t+1} := A\hat{x}_{t+1} - b \in \partial_{\hat{\varepsilon}_{t+1}}\theta(\hat{p}_{t+1})$.

PASSO 3. [*Medida de Melhora*]

Computar o "erro":

$$\hat{\varepsilon}_{t+1} := \alpha(1-\alpha)\,(\bar{v}_{t+1} - \hat{v}_t)^T(\hat{p}_t - \bar{\pi}_{t+1}) \;+\; (1-\alpha)\hat{\varepsilon}_t\,.$$

Computar a *medida de melhora* como em (25):

$$\delta_{t+1} := \hat{v}_{t+1}^T\,(\bar{\pi}_{t+1} - \hat{\pi}_k) \;+\; \hat{\varepsilon}_{t+1}\,. \tag{33}$$

PASSO 4. [*Teste-sério*]

Efetuar o *teste-sério*:

$$\theta(\bar{\pi}_{t+1}) \geq \theta(\hat{\pi}_k) + \tau_s\,\delta_{t+1}\,, \tag{34}$$

e:

- se (34) não se verifica, declarar um *passo-nulo* ou *iteração vermelha*;
- se (34) verifica-se, calcular $d := \hat{v}_{t+1}^T\,\bar{v}_{t+1}$, e:
  - se $d < 0$, declarar uma *iteração-amarela*;
  - se $d \geq 0$, então:
    - se o *teste de parada sério* $\delta_{t+1} \leq \delta_{\min}$ verifica-se, PARAR;
    - caso contrário, declarar um *passo-sério* ou *iteração-verde* e atualizar:
      - o limite inferior: $z_{\text{LB}} := \bar{z}_{t+1}$;
      - k := k + 1;
      - o *centro de estabilidade*: $\hat{\pi}_k := \bar{\pi}_{t+1}$;
      - o *conjunto de passos-sérios*: $T_s := T_s \cup \{t\}$.

PASSO 5. [*Testes de Parada*]

Efetuar os testes de parada como em VA:

– *Teste de parada primal-dual*: se

$$\frac{|c^T \hat{x}_{t+1} - z_{\text{LB}}|}{|z_{\text{LB}}|} < \xi_{pd}\,,$$

PARAR;

– *Teste de parada de viabilidade*: se

$$\frac{\|\hat{v}_{t+1}\|}{m} < \xi_v\,,$$

onde m denota o número de linhas da matriz A, PARAR ;

– *Teste de parada de otimalidade*: se

$$\text{UB} - z_{\text{LB}} < 1\,,$$

PARAR;

– *Teste de parada de número máximo de iterações*: se

$$t > maxiters\,,$$

PARAR;

– *Teste de parada de tempo máximo de CPU*: se

$$\text{CPU}_{\text{time}} > maxtime\,,$$

PARAR.

PASSO 6. [***Loop***]

Fazer $t := t + 1$ e retornar ao PASSO 1A. □

Vale a pena ressaltar as seguintes características importantes de RVA:

- se o ponto candidato $\bar{\pi}_{t+1}$ não gera um acréscimo significativo no valor de θ, então o centro de estabilidade não se altera (***passo-nulo*** em RVA, chamado de *iteração-vermelha*, página 22, em VA), o que implica que o multiplicador não é suficientemente bom. Neste caso, itera-se com o centro de estabilidade anterior, até que um ponto candidato significativamente melhor seja obtido;

- por outro lado, se ponto candidato $\bar{\pi}_{t+1}$ gera um acréscimo significante, i.e, pelo menos uma fração τ_s do acréscimo predito δ_{t+1}, então o centro de estabilidade é atualizado (***passo-sério*** em RVA, chamado de *iteração-verde*, página 22, em VA);

- o *teste de parada sério* no PASSO 4 requerendo que $\delta_{t+1} \leq \delta_{\min}$ pode ser substituído pelos *testes de parada sérios-split*:

$$||\hat{v}_{t+1}||^2 \leq \delta_v^2 \quad \text{e} \quad \hat{\varepsilon}_{t+1} \leq \delta_\varepsilon\,, \tag{35}$$

onde δ_v e δ_ε são duas *tolerâncias de parada sérias* dadas. Ao combinar-se (30) e (33) obtém-se a expressão:

$$\delta_{t+1} = s_t ||\hat{v}_{t+1}||^2 + \hat{\varepsilon}_{t+1} .$$

Portanto, sempre que $s_t \delta_v^2 + \delta_\varepsilon \leq \delta_{\min}$, (35) implica a condição de parada no PASSO 4. Uma vantagem potencial de (35) é que não depende do tamanho de passo s_t, que pode tornar-se demasiado pequeno conforme t cresce. Esta vantagem será confirmada pelos resultados numéricos apresentados em § 2.5, quando se comparam os dois algoritmos para dois grupos de instâncias do problema de Steiner em grafos.

- os demais *testes de parada* do PASSO 5 são os idênticos aos descritos no PASSO 4 do algoritmo do volume, página 22

1.4.2.2 Convergência do Algoritmo do Volume Revisado

Antes de provar que a seqüência gerada por RVA converge, primeiro mostra-se que os $\hat{v}_{t+1} := A\hat{x}_{t+1} - b$ do PASSO 2C de RVA são os extragradientes especiais assumidos "disponíveis" no PASSO 1 de ε-ESG.

Teorema 1.2 *Suponha que os pontos iniciais* $(\hat{p}_0, \hat{\varepsilon}_0, \hat{v}_0) := (\bar{\pi}_0, 0, \bar{v}_0)$ *foram dados. Então, RVA gera as seqüências* $\{(\hat{p}_{t+1}, \hat{\varepsilon}_{t+1}, \hat{v}_{t+1})\}_{t+1 \geq 0}$ *satisfazendo:*

$$\begin{cases} \hat{p}_{t+1} := \alpha \bar{\pi}_{t+1} + (1-\alpha)\hat{p}_t \,, \text{ como em (32)} \\ \hat{\varepsilon}_{t+1} := \alpha(1-\alpha)(\bar{v}_{t+1} - \hat{v}_t)^T(\hat{p}_t - \bar{\pi}_{t+1}) + (1-\alpha)\hat{\varepsilon}_t \\ \hat{v}_{t+1} := \alpha \bar{v}_{t+1} + (1-\alpha)\hat{v}_t \,, \; t \geq 0 \,. \end{cases}$$

Além disso, $\hat{v}_{t+1} = A\hat{x}_{t+1} - b$, *com* $\hat{x}_{t+1}$ *dado por* (31) *e* $\hat{v}_{t+1} \in \partial_{\hat{\varepsilon}_{t+1}} \theta(\hat{p}_{t+1})$ *para todo* $t+1 \geq 0$.

Prova: Para ver que $\hat{v}_{t+1} = A\hat{x}_{t+1} - b$, basta relembrar (31) e a definição de $\bar{v}_{t+1}$ no PASSO 1B de RVA.

Para mostrar que $\hat{v}_{t+1} \in \partial_{\hat{\varepsilon}_{t+1}} \theta(\hat{p}_{t+1})$, procede-se por indução sobre $t+1 \geq 0$:

Base. Quando $(t+1) = 0$, tem-se que $\hat{x}_0 = \bar{x}_0$, $\hat{p}_0 = \bar{\pi}_0$ e $\hat{v}_0 = \bar{v}_0$. Logo, $\hat{v}_0 = A\hat{x}_0 - b \in \partial\theta(\hat{p}_0)$. Ao relembrar que $\hat{\varepsilon}_0 = 0$, o resultado segue.

Hipótese. Para $(t+1) \geq 1$, fazer a hipótese indutiva de que $\hat{v}_t \in \partial_{\hat{\varepsilon}_t} \theta(\hat{p}_t)$.

Passo. Um vez que $\hat{v}_{t+1} = \alpha\bar{v}_{t+1} + (1-\alpha)\hat{v}_t$, onde $\bar{v}_{t+1} \in \partial\theta(\bar{\pi}_{t+1})$, ao aplicar (8) e (9), obtém-se, para todo $\pi \in \mathbb{R}^m$, que:

$$\begin{array}{rcl} \theta(\pi) & \leq & \theta(\bar{\pi}_{t+1}) + \bar{v}_{t+1}^T(\pi - \bar{\pi}_{t+1}) \quad \text{(a)} \\ \theta(\pi) & \leq & \theta(\hat{p}_t) + \hat{v}_t^T(\pi - \hat{p}_t) + \hat{\varepsilon}_t \quad \text{(b)}. \end{array}$$

A combinação convexa α (a) + $(1-\alpha)$ (b) das duas desigualdades acima gera, para todo $\pi \in \mathbb{R}^m$:

$$\theta(\pi) \leq \theta(\hat{p}_{t+1}) + \hat{v}_{t+1}^T\pi - (\alpha\bar{v}_{t+1})^T\bar{\pi}_{t+1} - ((1-\alpha)\hat{v}_t)^T\hat{p}_t + (1-\alpha)\hat{\varepsilon}_t\,.$$

Seja $\hat{\varepsilon}_{t+1}$ o termo para o qual a desigualdade acima pode ser escrita como:

$$\theta(\pi) \leq \theta(\hat{p}_{t+1}) + \hat{v}_{t+1}^T(\pi - \hat{p}_{t+1}) + \hat{\varepsilon}_{t+1}\,.$$

Então:

$$\begin{array}{rcl} \hat{\varepsilon}_{t+1} & := & \hat{v}_{t+1}^T\hat{p}_{t+1} - (\alpha\bar{v}_{t+1})^T\bar{\pi}_{t+1} - ((1-\alpha)\hat{v}_t)^T\hat{p}_t + (1-\alpha)\hat{\varepsilon}_t \\ & = & (\alpha\bar{v}_{t+1})^T(\hat{p}_{t+1} - \bar{\pi}_{t+1}) + ((1-\alpha)\hat{v}_t)^T(\hat{p}_{t+1} - \hat{p}_t) + (1-\alpha)\hat{\varepsilon}_t\,. \end{array}$$

Com base em (32) pode-se escrever:

$$\hat{p}_{t+1} - \bar{\pi}_{t+1} = (1-\alpha)\,(\hat{p}_t - \bar{\pi}_{t+1}) \quad \text{e} \quad \hat{p}_{t+1} - \hat{p}_t = \alpha\,(\bar{\pi}_{t+1} - \hat{p}_t)\,.$$

Utilizando-se essas igualdades na expressão para $\hat{\varepsilon}_{t+1}$ acima, obtém-se:

$$\begin{array}{rcl} \hat{\varepsilon}_{t+1} & = & (\alpha\bar{v}_{t+1})^T\,((1-\alpha)\,(\hat{p}_t - \bar{\pi}_{t+1})) + ((1-\alpha)\hat{v}_t)^T\,(\alpha\,(\bar{\pi}_{t+1} - \hat{p}_t)) \\ & & + (1-\alpha)\hat{\varepsilon}_t \\ & = & \alpha(1-\alpha)\,(\bar{v}_{t+1} - \hat{v}_t)^T(\hat{p}_t - \bar{\pi}_{t+1}) + (1-\alpha)\hat{\varepsilon}_t\,. \end{array}$$

Finalmente, aplica-se a Proposição 1.1 com $(\bar{\pi}, \bar{v})$ e $(\hat{p}, \hat{\varepsilon}, \hat{v})$ substituídos por $(\bar{\pi}_{t+1}, \bar{v}_{t+1})$ e $(\hat{p}_t, \hat{\varepsilon}_t, \hat{v}_t)$, respectivamente, para escrever:

$$\hat{\varepsilon}_{t+1} \geq -\alpha(1-\alpha)\hat{\varepsilon}_t + (1-\alpha)\hat{\varepsilon}_t = (1-\alpha)^2\,\hat{\varepsilon}_t\,,$$

uma quantidade não-negativa, por construção. □

O próximo passo consiste em provar a convergência global de RVA:

Teorema 1.3 *Seja* (1) *tal que o poliedro* Ψ *de* (4) *é limitado. Considere RVA com as tolerâncias de parada* (δ_{min} *ou* δ_v, *e* δ_ε) *inicializadas em* 0. *Suponha que o tamanho de passo* s_t *em* (20) *é escolhido de tal forma que* (27) *verifica-se para todo* $t+1 \geq 0$. *Se infinitos passos-sérios são relizados, então a seqüência* $\{\hat{p}_{t+1}\}_{t+1\in T_s}$ *possui um ponto de acumulação* $\hat{p}^\star$ *que resolve o problema dual* (5), *para* θ *definida em* (10).

Prova: Para poder aplicar o Corolário 1.1, necessita-se provar que a seqüência infinita $\{\hat{p}_{t+1}\}_{t+1\in T_s}$ é limitada. A hipótese sobre o conjunto viável Ψ implica que $\bar{x}_{t+1}$, um minimizador em (10) com $\bar{\pi} = \bar{\pi}_{t+1}$, é limitado para todo $t+1$, e também são limitados $\hat{x}_{t+1}$ e $\hat{v}_{t+1} = A\hat{x}_{t+1} - b$. Relembrando que, por (27), o tamanho de passo está limitado por cima e por baixo, basta ver a equação de deslocamento de extragradiente (30) para concluir que $\{\bar{\pi}_{t+1}\}_{t+1}$ é limitada. Portanto, $\{\hat{p}_{t+1}\}_{t+1\in T_s}$ é uma seqüência limitada também. □

Observação 1.8 Agora está-se em posição de mostrar como transformar RVA em um algoritmo de feixes, contanto que uma avaliação extra da função dual $\theta(\hat{p}_{t+1})$ seja feita:

- o conhecimento de $\theta(\hat{p}_{t+1})$ permite computar

$$\varepsilon_{t+1} := \hat{\varepsilon}_{t+1} + \theta(\hat{p}_{t+1}) + \hat{v}_{t+1}^T(\hat{\pi}_k - \hat{p}_{t+1}) - \theta(\hat{\pi}_k) \quad \text{tal que} \quad \hat{v}_{t+1} \in \partial_{\varepsilon_{t+1}}\theta(\hat{\pi}_k)\,,$$

  onde ε_{t+1} é não-negativo, porque $\hat{v}_{t+1} \in \partial_{\hat{\varepsilon}_{t+1}}\theta(\hat{p}_{t+1})$;

- sendo o parâmetro α variante com respeito a t, pode-se escolhê-lo de tal forma que $\hat{v}_{t+1} = \alpha_{t+1}\bar{v}_{t+1} + (1-\alpha_{t+1})\hat{v}_t$ satisfaça a condição de otimalidade para que $\bar{\pi}_{t+1}$ de (30) seja uma solução do seguinte problema:

$$\max_{p\in\mathbb{R}^m} \hat{\theta}_{t+1}(p) - \frac{1}{2s_t}\|p - \hat{\pi}_k\|^2\,, \tag{36}$$

  onde

$$\hat{\theta}_{t+1}(p) := \min\{\theta(\bar{\pi}_{t+1}) + \bar{v}_{t+1}^T(p - \bar{\pi}_{t+1})\,,\ \theta(\hat{p}_t) + \hat{v}_t^T(p - \hat{p}_t) + \hat{\varepsilon}_t\}\,.$$

  A função $\hat{\theta}_{t+1}$ é na realidade um modelo de planos de corte [18, Capítulo II] para a função côncava θ: ela é a escolha *minimal* de exatamente duas peças afins: aquela gerada por último (associada ao ponto candidato $\bar{\pi}_{t+1}$) e a peça *agregada* $\theta(\hat{p}_t) + \hat{v}_t^T(p - \hat{p}_t) + \hat{\varepsilon}_t$. Esta peça agregada possui um papel crucial na eliminação de elementos no feixe, sem no entanto prejudicar a convergência, o que pode ser visto em [42]. Essa expressão para uma função convexa pode ser também encontrada nas equações (4.1) de [24] e (7.17) de [18];

- ao escrever-se o dual de (36) obtém-se a relação:

$$\hat{\theta}_{t+1}(\bar{\pi}_{t+1}) = \theta(\hat{\pi}_k) + s_t\|\hat{v}_{t+1}\|^2 + \varepsilon_{t+1}\,.$$

Então, fazendo com que a medida de melhora δ de (25) seja agora dada por:

$$\delta'_{t+1} := s_t\|\hat{v}_{t+1}\|^2 + \varepsilon_{t+1},$$

o teste-sério (26) torna-se:

$$\theta(\bar{\pi}_{t+1}) \geq \theta(\hat{\pi}_k) + \tau_s(\hat{\theta}_{t+1}(\bar{\pi}_{t+1}) - \theta(\hat{\pi}_k))\,,$$

um teste-sério padrão para algoritmos de feixes.

Ao todo, obtém-se a chamada forma penalizada dos métodos de feixes, implementada com compressão máxima de feixes a cada iteração. □

1.4.2.3 Recuperação Primal

Nesta subseção estuda-se a recuperação primal, em particular introduz-se um limite *a posteriori* para o erro de aproximação relacionando o ponto estimado primal gerado por RVA com uma solução primal.

Se as *tolerâncias de parada sérias* $\delta_{\min}$ ou δ_v e δ_ε e as demais *tolerâncias de parada* ξ_{pd} e ξ_v são inicializadas em 0 no PASSO 0 de RVA, e se os demais testes de parada do PASSO 5 de RVA são desativados, o algoritmo entra em *loop*. Quando T_s é infinito, mostrou-se no Teorema 1.3 que $\{\hat{p}_{t+1}\}_{t+1\in T_s}$, $t+1 \geq 0$ é uma seqüência maximizadora. Seja $\hat{p}^\star$ um ponto limite resolvendo (5). Então uma solução primal, i.e., a solução de (1), é dada por:

$$x(\hat{p}^\star) \in \operatorname*{Argmin}_{x\in\Psi} L(x,\hat{p}^\star)\,,\ \text{tal que}\ Ax(\hat{p}^\star) = b\,.$$

Quando o *teste de parada sério* ou os *testes de parada sérios-split* do PASSO 4 e os demais *testes de parada* do PASSO 5 de RVA são reativados, existe um último passo-sério, diga-se $k_{\mathtt{last}}$, correspondendo a $t_{\mathtt{last}} \in T_s$. O resultado a seguir estabelece um limite *a posteriori* para o erro de aproximação relacionando o ponto estimado primal $\hat{x}_{t_{\mathtt{last}}}$ de (31) com uma solução primal.

Teorema 1.4 *Considere RVA com os testes de parada sérios-split* (35) *utilizados no* PASSO 4, *e com as tolerâncias de parada sérias* $\delta_v, \delta_\varepsilon > 0$ *fornecidas no* PASSO 0. *Seja* $(t+1)^*$ *o índice tal que a parada ocorre no* PASSO 4 *e seja* $(\hat{x}_{\texttt{last}}, \hat{v}_{\texttt{last}})$ *o par* $(\hat{x}_{t+1}, \hat{v}_{t+1})$ *gerado por último. Então, existe* $L > 0$ *tal que:*

$$\|x^\star - \hat{x}_{\texttt{last}}\| \leq L\delta_v\,,$$

para algum $x^\star$ *resolvendo* (1).

Prova: Por construção, $\hat{x}_t$ pertence ao poliedro Ψ. Para todo $x \in \Psi$ tal que $Ax - b = A\hat{x}_{\texttt{last}} - b$, a relação de dualidade fraca (7) implica que $\hat{x}_{\texttt{last}}$ resolve o problema:

$$\left\{\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & c^T x \\ Ax - b & = A\hat{x}_{\texttt{last}} - b \\ Dx & = e \\ x & \geq 0\,. \end{array}\right. \tag{37}$$

Observe que (37) corresponde a uma perturbação no lado direito do problema primal (1). Pelo Teorema 2.4 de [51], soluções de problemas de programação linear são *Lipschitz*-contínuas em relação a perturbações no lado direito:

$$\|x^\star - \hat{x}_{\texttt{last}}\| \leq L\,\|A\hat{x}_{\texttt{last}} - b\| = L\,\|\hat{v}_{\texttt{last}}\|\,.$$

Juntando-se isso às expressões dos *testes de parada sérios-split* (35), obtém-se o resultado. □

Este resultado é conhecido como *O Teorema de Everett* e pode ser encontrado em [32] e [37, Capítulo XII, Teorema 2.1.1].

Parte II

Resolução do Problema de Steiner em Grafos com o Algoritmo do Volume

2 Resolução do Problema de Steiner em Grafos com o Algoritmo do Volume

Nesta seção apresenta-se um novo algoritmo para a resolução do problema de Steiner em grafos, baseado no algoritmo do volume [5].

Dado um grafo não-direcionado $G = (V, E)$ e um subconjunto $T \subseteq V$ de vértices ditos *terminais*, uma *árvore de Steiner* é uma árvore que gera T. Seja c_e, para cada $e \in E$, um *custo* não-negativo associado à aresta do grafo. O *Problema de Steiner em Grafos* (SPG) consiste em encontrar a *árvore de Steiner de custo mínimo*, onde o custo de uma árvore é dado pela soma dos custos das suas arestas.

Este problema de otimização é sabido *NP*-difícil [40]. Na verdade, o problema é ainda *NP*-difícil mesmo quando os grafos são bipartidos [40], cordais [70], fortemente cordais [70], cordais bipartidos [54], *split* [70], planares [34] ou completos com custos iguais a 1 ou 2 [15].

O problema de Steiner em grafos tem sido largamente aplicado ao desenvolvimento de sistemas de comunicação, distribuição e transporte. Muitos problemas de desenho de redes podem ser formulados como um problema de Steiner em grafos, direcionados ou não, dependendo da aplicação. Pode-se citar, por exemplo, *uncapacitated plant location* [49, 50] e *network design* [50]. Motivada pela crescente demanda das redes de telecomunicação, a solução do problema de Steiner tem recebido atenção considerável nos últimos anos.

Para o problema de Steiner em grafos existem na literatura distintas formulações como problemas de programação inteira, assim como diversas técnicas de resolução e de redução do tamanho do grafo (técnicas de pré-processamento). Existem também estudos sobre casos particulares que admitem solução polinomial (grafos série-paralelo [69, 58, 59], k-planares [13, 14], fortemente cordais com custos iguais a 1 [70], etc.) e variações do problema de Steiner, como por exemplo, o problema de Steiner *prize-collecting* e o problema de Steiner com restrições de grau sobre os vértices, entre outros. Uma referência bastante completa abrangendo esses tópicos é o livro *The Steiner Tree Problem* [38].

O problema de Steiner foi muito estudado por pessoas de várias áreas diferentes, devido ao seu largo espectro de aplicações. Por isso, há muitas referências sobre esse problema na literatura. Ao longo deste texto, portanto, citam-se aquelas que foram consideradas mais relevantes.

Nos trabalhos de Maculan [49] e Goemans & Myung [35] encontram-se diferentes formulações de programação inteira para o problema, e em [35] é mostrada a equivalência entre algumas dessas formulações.

Existem na literatura diversas técnicas de resolução. A seguir citam-se algumas delas: algoritmos aproximativos [12, 71]; *branch-and-cut* [20, 21, 22, 48, 43] (este último [43] o mais recente e completo sobre o problema); relaxação lagrangeana e desigualdades válidas [47]; redes neurais [57]; meta-heurísticas (algoritmos genéticos [31, 39], *simulated annealing* [27], busca-tabu [29, 77, 78, 61], GRASP [52, 53]; entre outras). Foram desenvolvidas também heurísticas específicas para o problema [67, 68, 73, 30], dentre as quais destaca-se a de Takahashi e Matsuyama [67], que até hoje é a que possui a melhor relação tempo/qualidade, sendo utilizada como base para muitas das meta-heuríscas citadas anteriormente e como heurística primal básica em vários algoritmos exatos.

Foram estudadas algumas técnicas de redução para o tamanho do problema. A mais conhecida é a descrita em [28], baseada em testes de sensibilidade sobre os custos reduzidos gerados por relaxações lineares do problema. Essa técnica foi utilizada com sucesso nos algoritmos mais recentes de *branch-and-cut* [48, 43].

Um caso particular importante do problema de Steiner em grafos dá-se quando o grafo é um reticulado, devido basicamente à sua aplicação ao desenho de circuitos eletrônicos integrados. O problema de Steiner sobre um reticulado é considerado na literatura como um dos mais difíceis entre os problemas de Steiner em grafos. Devido à sua simetria, torna-se um problema com grande multiplicidade de soluções, o que dificulta bastante a sua resolução. O reticulado em questão pode ter ou não buracos, que modelam os obstáculos dos problemas reais de desenho de circuitos VLSI. A existência de buracos torna o problema ainda mais difícil. Por isso, instâncias desse tipo foram escolhidas nesta Tese como a principal fonte de testes para a validação do algoritmo do volume para problemas de Steiner em grafos.

A título de curiosidade, o problema de Steiner em grafos é na realidade uma versão combinatória do *Problema de Steiner Euclideano*, que originou-se no início do século $XVII$ quando *Fermat* propôs o seguinte problema: *dados três pontos no plano, encontrar um quarto ponto tal que a soma das distâncias desse ponto aos três pontos dados seja mínima.* Esse problema é conhecido na literatura [38] como *O Problema de Fermat.* Toricelli propôs uma solução geométrica para esse problema antes de 1640. F. Heinen, aparentemente, foi o primeiro a provar em 1834 que, para um triângulo em que um ângulo é maior ou igual a 120°, o vértice desse ângulo é o ponto minimizador [49]. O *Problema Geral de Fermat* refere-se ao problema de encontrar no plano um ponto tal que a soma das distâncias deste ponto a n pontos dados seja mínima [45]. Jacob Steiner foi o mais famoso representante de geometria da universidade de Berlin durante o século XIX e trabalhou muito no problema. O problema de Fermat foi

popularizado em [25] sob o nome de *problema de Steiner*. O *Problema de Steiner Euclideano* consiste em encontar uma árvore que gere n pontos dados no plano Euclideano a mínimo custo. Uma árvore mínima que gere esses n pontos é chamada de *árvore de Steiner* e pode conter outros pontos (chamados *pontos de Steiner*) que não aqueles dados anteriormente.

Esta seção é organizada da seguinte maneira: em § 2.1 apresenta-se a formulação de programação linear inteira utilizada; em § 2.2 mostra-se como a relaxação Lagrangena é aplicada àquela formulação, em § 2.2.1 como resolve-se o problema dual Lagrangeano, e em § 2.2.2 como resolvem-se os subproblemas Lagrangeanos; em § 2.3 apresentam-se as heurísticas primais desenvolvidas nesta Tese; em § 2.4 são exibidos os resultados computacionais derivados da aplicação do algoritmo do volume a alguns problemas da *SteinLib* [43]; finalmente, em § 2.5 são exibidos alguns testes computacionais que comparam as performances dos algoritmos do volume e do volume revisado.

2.1 Formulação

Dado um grafo direcionado $D = (V, A)$, um conjunto de terminais $T \in V$ e um vértice raiz $s \in T$, uma *arborescência de Steiner* é uma árvore direcionada tendo como raiz o vértice s e gerando os vértices em T, i.e., uma árvore de Steiner direcionada. O *problema da arborescência de Steiner* (SAP) consiste em encontrar a arborescência de Steiner de custo total mínimo. Não é difícil de ver que qualquer instância do problema de Steiner em grafos (SPG) pode ser equivalentemente formulada como um problema da arborescência de Steiner (SAP) bidirecionado. Foi mostrado em [21] que ao utilizar-se a formulação bidirecionada do SPG, obtém-se uma melhor relaxação do que aquela dada pela formulação não-direcionada.

Neste trabalho o SAP bidirecionado é formulado como um problema de fluxo não-simultâneo de mercadorias (*nonsimultaneous multi-commodity flow problem*) sobre um grafo direcionado $D = (V, A)$. Este grafo direcionado possui o mesmo conjunto de vértices do grafo original $G = (V, E)$ e seu conjunto de arcos A é obtido da seguinte maneira: para cada aresta $e = [i, j] \in E$ são criados arcos $(i, j) \in A$ e $(j, i) \in A$ de tal forma que $c_{ij} = c_{ji} = c_e$.

A formulação detalhada a seguir foi introduzida simultaneamente por Claus & Maculan [23] e Wong [76]. Um vértice terminal $s \in T$ é escolhido para ser a *fonte* ofertando $|T_0| := |T \setminus \{s\}|$ mercadorias, cada mercadoria demandada por cada um dos restantes $|T_0|$ vértices terminais. As variáveis f_{ij}^k, para $(i, j) \in A$ e $k = 1, \ldots, T_0$, indicam a quantidade de mercadoria k transportada pelo arco (i, j). As variáveis binárias x_{ij}, para $(i, j) \in A$, controlam a inclusão ($x_{ij} = 1$) ou não ($x_{ij} = 0$) do arco

$(i,\vec{j})$ na solução. Denotando por $I(i)$ o conjunto de vértices $j \in V$ tais que $(j,\vec{i}) \in A$ e por $O(i)$ o conjunto de vértices $j \in V$ tais que $(i,\vec{j}) \in A$, o problema da arborescência de Steiner pode ser formulado como segue:

$$
\begin{cases}
\min\limits_{x:=(x_{ij})_{(i,\vec{j})\in A}} \quad c^T x & \\
\sum\limits_{j\in O(s)} f^k_{sj} - \sum\limits_{j\in I(s)} f^k_{js} = 1\,, \quad k \in T_0 & (a_1) \\
\sum\limits_{j\in O(k)} f^k_{kj} - \sum\limits_{j\in I(k)} f^k_{jk} = -1\,, \quad k \in T_0 & (a_2) \\
\sum\limits_{j\in O(i)} f^k_{ij} - \sum\limits_{j\in I(i)} f^k_{ji} = 0\,, \quad i \in V\setminus\{s,k\}, \quad k \in T_0 & (a_3) \\
x_{ij} \in \{0,1\}\,, \quad (i,\vec{j}) \in A\,. & (a_4) \\
0 \le f^k_{ij} \le x_{ij}\,, \quad (i,\vec{j}) \in A\,, \quad k \in T_0\,. & (a_5)
\end{cases}
\tag{38}
$$

As equações (38)(a_1)-(a_3) representam o balanço de mercadorias em cada vértice. Para a fonte esse balanço é positivo (oferta) e unitário em relação a cada mercadoria, para cada terminal em T_0 o balanço é negativo (demanda) e unitário com respeito àquela mercadoria relacionada a ele, e para os demais vértives não-terminais em V esse balanço é nulo (tudo o que é recebido é enviado). As restrições (a_4) denotam a integralidade das variáveis de decisão x_{ij}. As restrições (a_5) representam os limites de fluxo para cada arco; repare que só é possível passar fluxo através de um arco $(i,\vec{j})$ quando este faz parte da árvore de Steiner, ou seja, quando $x_{ij} = 1$. Finalmente, com a função objetivo deseja-se minimizar a soma dos custos dos arcos que fazem parte da árvore.

A última parte desta subseção é devotada a uma pequena análise sobre a relação entre as relaxações contínuas de duas formulações distintas de programação inteira para o SAP, aquela utilizada neste trabalho e a formulação *dicut* clássica utilizada em [43].

A formulação (38) possui um número polinomial de restrições e um número polinomial de variáveis. Seja P_{xf} o politopo definido por (38)(a_1)-(a_3), pela relaxação contínua de (38)(a_4) e por (38)(a_5). Mostra-se em [49] que o politopo P_x obtido pela projeção de P_{xf} sobre as variáveis x pode ser caracterizado como segue:

$$P_x = \{x : x(\delta^+(S)) \geq 1, \text{ para todo } S \subset V \text{ com } r \in S \text{ e } (V \backslash S) \cap T \neq \emptyset; \\ \text{e } 0 \leq x_{ij} \leq 1, (i,\vec{j}) \in A\},$$

onde $\delta^+(S) := \{(i,\vec{j}) \in A : i \in S, j \in V \backslash S\}$ e $x(\delta^+(S)) := \sum_{(i,\vec{j}) \in \delta^+(S)} x_{ij}$.

A *formulação dicut* para o SAP, baseada no politopo P_x e proposta em [2], é definida da seguinte maneira:

$$\begin{cases} \min\limits_{x:=(x_{ij})_{(i,\vec{j}) \in A}} & c^T x \\ x(\delta^+(S)) \geq 1, & \text{para todo } S \subset V, r \in S, (V \backslash S) \cap T \neq \emptyset \quad (a_6) \\ x_{ij} \in \{0,1\}, & (i,\vec{j}) \in A \end{cases} \tag{39}$$

Uma vez que P_x é a projeção de P_{xf} sobre as variáveis x, as relaxações contínuas de (38) e (39) são equivalentes. É importante mencionar que, mesmo possuindo um número exponencial de restrições, a relaxação contínua da formulação *dicut* (39) para o SAP pode ser resolvida em tempo polinomial com o algoritmo do elipsóide. Isso é possível porque o problema de separação sobre as *desigualdades dicut de Steiner* (39)(a_6) é polinomialmente resolvível por um algoritmo de *maxcut* [2].

Devido a seu enorme número de variáveis, a formulação de fluxo não-simultâneo de mercadorias (38) para o SAP praticamente não foi utilizada em algoritmos baseados na relaxação linear do problema. A abordagem por relaxação Lagrangeana, introduzida neste trabalho e detalhada a seguir, mostrou ser possível lidar-se computacionalmente com (38).

2.2 Relaxação Lagrangeana

As restrições mais difíceis de (38) são as relativas ao balanço de mercadorias, ou seja, (38)(a_1)-(a_3). Associa-se a cada restrição de balanço de mercadorias um multiplicador de Lagrange (ou variável dual) π_i^k, com $i \in V$ e $1 \leq k \leq |T_0|$, obtendo-se o *vetor de multiplicadores de Lagrange* $\pi = (\pi_i^k) \in \mathbb{R}^{|V||T_0|}$. Dualizando-se Lagrangeanamente as restrições (38)(a_1)-(a_3), e relaxando-se a restrição de integralidade (38)(a_4), obtém-se

o seguinte *subproblema Lagrangeano*:

$$\begin{cases} \min\limits_{\substack{x:=(x_{ij})_{(i,j)\in A} \\ f:=(f_{ij})_{(i,j)\in A}}} \quad c^T x + \sum\limits_{k\in T_0} {\ell^k}^T f^k + \xi(\bar{\pi}) \\ \\ \quad x_{ij} \in [0,1]\,, \quad (i,j)\in A\,. \qquad (a_4') \\ \\ \quad 0 \le f_{ij}^k \le x_{ij}\,, \quad (i,j)\in A\,, \quad k\in T_0\,, \quad (a_5) \end{cases} \tag{40}$$

onde:

- $\ell^k := (\ell_{ij}^k)$ é o *vetor de custos Lagrangeanos* associados às variáveis f_{ij}^k (custos estes que dependem dos multiplicadores de Lagrange π_i^k);
- o fator $\xi(\bar{\pi})$ é uma constante facilmente computada para qualquer valor fixo de $\bar{\pi} \subset \pi$, $|\bar{\pi}| = 2|T_0|$ (ou seja, $\bar{\pi}$ contém os índices de π que são utilizados na dualização Lagrangeana das restrições (38)(a_1)-(a_2)); e
- o objetivo é dado pela *função lagrangeana* $L(x, f^k, \pi^k)$:

$$L(x, f^k, \pi^k) := \langle c, x\rangle + \langle \ell^k, f^k\rangle + \xi(\pi^k)\,.$$

Fazendo-se a co-relação com o que foi apresentado em § 1.1 e § 1.2.1, as restrições (38)(a_1)-(a_3) fazem o papel das restrições (1)(a), e o poliedro Ψ definido em (4) é representado aqui pelas restrições (38)(a_4')-(a_5).

Conforme visto em § 1.2.1, cada vetor de multiplicadores de Lagrange $\bar{\pi}$ conduz a um *limite inferior* para o custo ótimo do problema (38). Logo, o interessante é encontrar o maior limite inferior possível, o que pode ser feito resolvendo-se o seguinte problema de programação linear, chamado de *problema dual*:

$$\max_{\pi\in\mathbb{R}^{|V||T_0|}} \theta(\pi)\,, \tag{41}$$

cujo objetivo é dado pela *função dual*:

$$\theta(\pi^k) := \min_{\substack{x\in(a_4') \\ f^k\in(a_5)}} L(x, f^k, \pi^k)\,, \tag{42}$$

para cada mercadoria k.

Nas subseções seguintes descrevem-se os valores dos parâmetros utilizados no algoritmo do volume para a resolução do problema dual (41)-(42) e mostra-se como os subproblemas (40) são resolvidos.

2.2.1 Resolução do Problema Dual Lagrangeano

Para resolver o problema dual (41)-(42) considera-se, tanto para VA (resultados computacionais em § 2.4) quanto para RVA (resultados computacionais em § 2.5), os seguintes parâmetros:

- como parâmetros iniciais no PASSO 0 ([*Inicializações*]):
  - vetor de multiplicadores de Lagrange: $\bar{\pi}_0 = 0 \in \mathbb{R}^m$,
  - limite inferior: $z_{\text{LB}} = -10^{31}$,
  - escalar: $\alpha = 0.001$,
  - tolerância-de-parada-primal-dual: $\xi_{pd} = 0.001$,
  - tolerância-de-parada-de-viabilidade: $\xi_v = 0.001$,
  - número máximo de iterações: $maxiters = 30000$,
  - limite de tempo de CPU: $maxtime = 10500$ segundos;

- como valores de atualização do fator de correção μ do tamanho de passo (20) no PASSO 1A ([*Deslocamento de Extragradiente*]) :
  - $\mu = 0.67\,\mu$, a cada 20 *iterações-vermelhas*,
  - $\mu = 1.10\,\mu$, a cada 400 *iterações-amarelas*,
  - $\mu = 2.00\,\mu$, a cada *iteração-verde*;

- como parâmetros de atualização de $\alpha_{\max}$, parâmetro utilizado para atualizar o escalar α:
  - $\alpha_{\text{factor}} = 0.5$,
  - $\alpha_{\text{int}} = 250$ iterações.

2.2.2 Resolução dos Subproblemas Lagrangeanos

Os subproblemas Lagrangeanos (40) podem ser facilmente resolvidos por inspeção. Para cada arco $(i,\vec{j}) \in A$:

- Calcula-se $\sum_{k:\ell_{ij}^k<0} |\ell_{ij}^k|$.

  - Se $\sum_{k:\ell_{ij}^k<0} |\ell_{ij}^k| > c_{ij}$, então:
    * $x_{ij} = 1$,
    * $f_{ij}^k = 1$, para todo k tal que $\ell_{ij}^k < 0$,
    * $f_{ij}^k = 0$, para todo k tal que $\ell_{ij}^k \geq 0$;

  - Se $\sum_{k:\ell_{ij}^k<0} |\ell_{ij}^k| \leq c_{ij}$, então:
    * $x_{ij} = 0$,
    * $f_{ij}^k = 0$, para todo k.

2.3 Heurísticas Primais

Nesta subseção são descritas as três heurísticas primais desenvolvidas nesta Tese para encontrar limites superiores para o problema (38).

2.3.1 Árvore Geradora Mínima com Custos "Volumétricos"

Essa primeira heurística primal, chamada de *MSTV* em § 2.4, consiste em: a cada iteração t do algoritmo do volume definem-se os custos "volumétricos" de cada arco $(i,\vec{j})$ da seguinte maneira:

- c_{ij}, para $t = 0$;
- $-\hat{x}_{ij}$, para $t > 0$,

onde $\hat{x}_{ij}$ é a aproximação primal (21) para cada arco $(i,\vec{j}) \in A$.

A seguir encontra-se uma árvore geradora mínima dos vértices do grafo direcionado $G_d = (V, E_d)$ com os custos acima definidos, e finalmente podam-se as folhas que não são vértices terminais.

Motivação. Repare que minimizar $-\hat{x}_{ij}$ é o mesmo que maximizar $\hat{x}_{ij}$. Então a idéia é dar maior peso para aqueles arcos que tenham valor mais alto na estimativa primal $\hat{x}$ gerada pelo algoritmo do volume. Relembrando o que foi exposto em § 1.3.2 quando

se falou da *interpretação geométrica* do Teorema do Volume 1.2, quando $\hat{x}_{ij}$ possui um valor alto é porque o arco $(i,\vec{j})$ participa de um grande número de soluções de subproblemas (40), o que, intuitivamente, aumentaria sua probabilidade de participar de uma solução do problema original (38).

2.3.2 Árvore Geradora Mínima Modificada com Custos "Volumétricos"

Essa segunda heurística primal, chamada de $MMSTV$ em § 2.4 consiste no seguinte: a cada iteração t do algoritmo do volume definem-se os custos "volumétricos" de cada arco $(i,\vec{j})$ da seguinte maneira:

- c_{ij}, para $t = 0$;
- $(1 - \hat{x}_{ij})\, c_{ij}$, para $t > 0$.

Seja β um parâmetro com valor inicial 0.6. Seja V' o subconjunto dos vértices do grafo direcionado $G_d = (V, E_d)$ formado pelos vértices terminais mais todos os vértices não-terminais tais que

$$\sum_{j:(i,j)\in E_d} \hat{x}_{ij} \geq \beta .$$

Se o subgrafo induzido pelos vértices de V' for conexo, procura-se uma árvore geradora mínima com os custos acima definidos, e finalmente podam-se as folhas que não são vértices terminais.

Caso o subgrafo anteriormente mencionado seja desconexo, subtrai-se sucessivamente do β anterior o valor 0.1, redefinindo-se um novo subconjunto de vértices V' em função de cada novo β, até encontrar-se um V' que seja conexo. Feito isso, procura-se uma árvore geradora mínima com os custos "volumétricos" acima definidos, e finalmente podam-se as folhas que não são vértices terminais.

Motivação. Dar prioridade tanto aos arcos que possuam valor alto na solução primal estimada $\hat{x}$, quanto aos vértices não-terminais cujos arcos incidentes possuam valor alto na solução primal estimada $\hat{x}$. Repare que nesta heurística está sendo mais aproveitada a informação primal fornecida pelas soluções estimadas pelo algoritmo do volume.

2.3.3 Heurística de Takahashi & Matsuyama com Custos "Volumétricos"

Essa terceira e última heurística primal, chamada de $T\&MV$ em § 2.4, é essencialmente igual àquela definida por Takahashi & Matsuyama em [67], com algumas idéias de randomização propostas em [60] e custos baseados na solução estimada produzida pelo algoritmo do volume.

A idéia dessa heurística consiste em começar com um vértice terminal e conectá-lo ao vértice terminal que possua caminho mais curto ao terminal inicial. O vértice terminal seguinte é escolhido entre os terminais restantes de tal maneira que ele esteja o mais próximo possível do caminho ou subárvore existente até o momento. Esse processo continua até que todos os terminais sejam conectados.

Para o cálculo dos caminhos mais curtos acima mencionados são utilizados os seguintes custos "volumétricos" para os arcos (i,j):

- c_{ij}, para $t = 0$;
- $(1 - \hat{x}_{ij})\, c_{ij}$, para $t > 0$.

Como sugerido em [60] e utilizado também em [43], tenta-se melhorar a heurística encontrando a árvore geradora mínima dos vértices gerados por $T\&M$ e podando as folhas que não são vértices terminais.

A escolha do vértice terminal inicial é feita da seguinte maneira:

- na iteração inicial, e a cada φ iterações do algoritmo do volume selecionam-se randomicamente 10 vértices terminais, computa-se a árvore de Takahashi & Matsuyama (com os custos "volumétricos" definidos anteriormente) começando com cada um desses terminais. Ao final desse processo, guarda-se o terminal que gerou a melhor solução (menor limite superior);
- nas demais iterações começa-se sempre com aquele vértice terminal guardado anteriormente.

Essa idéia, aqui adaptada ao algoritmo do volume, foi utilizada também por Koch & Martin em [43], onde o parâmetro φ é fixo: a seleção randômica é feita a cada 5 iterações de geração de planos de corte. Neste trabalho este parâmetro depende do tamanho da instância, o que será visto com mais detalhes em § 2.4.

Observação 2.1 Em § 2.4 e nos Apêndices 1, 2 e 3 pode ser observado que a "intuição" de utilizar, nas três heurísticas, o valor da estimativa primal $\hat{x}_{ij}$ dada pelo algoritmo do volume para modificar os custos dos arcos tem um impacto muito positivo, vista a qualidade dos limites superiores gerados.

2.4 Resultados Computacionais

Nesta seção são apresentados os resultados computacionais relativos à aplicação do algoritmo do volume, página 21, ao problema de Steiner em grafos. Esses resultados são comparados com os apresentados por T. Koch e A. Martin em [43].

No trabalho de T. Koch e A. Martin é proposto um algoritmo do tipo *branch-and-cut* para a resolução da formulação *dicut* (39). São utilizadas também várias técnicas de pré-processamento do tamanho do grafo e a heurística de Takahashi & Matsuyama modificada pelas soluções contínuas fornecidas ao longo do processo. O algoritmo proposto em [43] é conhecido na literatura como o melhor para a resolução do problema de Steiner em grafos.

O código para o algoritmo desenvolvido nesta Tese foi implementado em conjunto com o Dr. Francisco Barahona, em C++ e todas as instâncias foram rodadas em uma estação de trabalho RSI6000 com processador de 332MHz.

As instâncias foram extraídas da biblioteca *SteinLib*, disponível por *ftp* anônimo no endereço `ftp://ftp.zib.de/pub/mp-testdata/SteinLib/` ou na seguinte *url*: `http://www.zib.de/pub/mp-testdata/SteinLib/`.

Os conjuntos de problemas testados incluem as seguintes séries:

- B, C, D e E: problemas-teste introduzidos por J. Beasley [11], definidos sobre grafos randômicos esparsos com custos euclidianos;
- R: problemas-teste introduzidos por J. Soukup e W. F. Chow [66], definidos sobre reticulados sem buracos;
- ALUE, ALUT, DIW, DMXA, GAP, MSM e TAQ: problemas reais derivados de circuitos eletrônicos VLSI, introduzidos por T. Koch e A. Martin [43], e definidos sobre reticulados com buracos (obstáculos).

Os valores dos parâmetros de inicialização do algoritmo já foram descritos em § 2.2.1. Para as séries de problemas B, C, D, E e R as heurísticas $MSTV$ (árvore geradora mínima com custos volumétricos) e $MMSTV$ (árvore geradora mínima modificada com custos volumétricos) são executadas a cada 20 iterações; já a heurística $T\&MV$ (heurística de Takahashi & Matsuyama com custos volumétricos) é executada a cada 200 ou 500 (parâmetro φ definido em § 2.3.3) iterações, dependendo do tamanho do problema. Para as demais instâncias testadas (VLSI) as heurísticas $MSTV$ e $MMSTV$ são executadas a cada 100 ou 200 iterações, e a heurística $T\&MV$ é executada a cada 1000, 2500 ou 5000 iterações, dependendo do tamanho do pro-

Conclusões e Extensões

blema. Conforme descrito em § 2.2.1, a execução do algoritmo é interrompida quando é atingido o tempo máximo de CPU de 10500 segundos, desde que nenhum dos demais critérios de parada tenham sido satisfeitos.

Os resultados completos são apresentados nas tabelas dos Apêndices 1, 2 e 3. Cada linha de cada tabela é relativa a uma instância. Na primeira coluna (*Nome*) aparece o nome da instância. Nas três colunas seguintes ($|V|, |E|$ e $|T|$) é descrito o tamanho da instância: número de vértices, arestas e vértices terminais, respectivamente. Na quarta coluna (LB) aparece o melhor limite inferior encontrado pelo algoritmo. Na coluna seguinte (UB) aparece o melhor limite superior encontrado pelas heurísticas primais. Na sexta coluna ($Gap(\%)$) aparece o salto de dualidade em percentagem, definido por $100\,\frac{UB-LB}{LB}$. Na coluna seguinte ($CPU_{\text{Tempo}}(s)$) aparece o tempo total de CPU, decorrido até que algum dos critérios de parada seja satisfeito ou o limite máximo de tempo seja atingido. As três seguintes colunas ($MSTV, MMSVT$ e $T\&MV$) exibem os melhores limites superiores encontrados por cada uma das heurísticas primais implementadas. Na última coluna (PC) é apresentada uma comparação da performance do algoritmo desenvolvido nesta Tese com o apresentado em [43], sem levar em conta os tempos de CPU:

- $*^{+}$: indica que a otimalidade foi provada pelo algoritmo desenvolvido nesta Tese e não foi provada pelo algoritmo apresentado em [43];
- $*^{-}$: indica que a otimalidade foi provada pelo algoritmo apresentado em [43] e não foi provada pelo algoritmo desenvolvido nesta Tese;
- $+$: indica que a otimalidade não foi provada por nenhum dos dois algoritmos, sendo melhor o salto de dualidade obtido pelo algoritmo desenvolvido nesta Tese;
- $-$: indica que a otimalidade não foi provada por nenhum dos dois algoritmos, sendo melhor o salto de dualidade obtido pelo algoritmo apresentado em [43];
- $\sim$: indica que os dois algoritmos apresentaram resultados equivalentes, i.e., ou ambos provaram otimalidade ou os saltos de dualidade obtidos foram iguais.

Devido ao grande número de instâncias testadas e o decorrente grande volume de informação gerado, optou-se por apresentar nesta seção uma única tabela resumida de resultados percentuais, comparando as performances do algoritmo desenvolvido nesta Tese e do algoritmo apresentado em [43] (utilizando os dados apresentados nas tabelas do artigo).

Na Tabela 1 a seguir há uma linha para cada série de instâncias. A primeira coluna (*Série*) indica o nome da série. A coluna seguinte (TS) indica o número de instâncias testadas em cada série. A terceira coluna ($OPT(\%)$) indica a percentagem das instâncias da série para as quais provou-se otimalidade. A coluna

seguinte ($Gap_m(\%)$) indica o salto de dualidade médio, em percentagem, para os problemas nos quais a otimalidade não é provada. As últimas cinco colunas ($*^+(\%)$, $+(\%)$, $*^-(\%)$, $-(\%)$ e $\sim(\%)$) possuem o mesmo significado dos símbolos descritos anteriormente para a última coluna das tabelas dos Apêndices; sendo aqui os resultados apresentados em percentagem, para cada série de problemas.

Série	*TS*	$OPT(\%)$	$Gap_m(\%)$	$*^+(\%)$	$+(\%)$	$*^-(\%)$	$-(\%)$	$\sim(\%)$
B	18	100.00	0.00	0.00	0.00	0.00	0.00	100.00
C	20	90.00	0.31	0.00	0.00	10.00	0.00	90.00
D	19	52.63	0.56	0.00	0.00	36.84	0.00	63.16
E	14	57.14	1.06	0.00	0.00	42.86	0.00	57.14
R	46	100.00	0.00	0.00	0.00	0.00	0.00	100.00
ALUE	13	23.08	7.00	0.00	7.69	23.08	38.46	30.77
GAP	13	84.61	14.04	0.00	0.00	15.39	0.00	84.61
TAQ	14	64.29	17.44	0.00	7.14	0.00	21.43	71.43
ALUT	07	42.86	6.05	0.00	42.86	0.00	14.28	42.86
DMXA	14	92.86	3.95	0.00	7.14	0.00	0.00	92.86
MSM	30	90.00	1.94	6.67	6.67	0.00	3.33	83.33
DIW	21	76.19	14.03	9.52	23.81	0.00	0.00	66.67

Tabela 1: Resultados percentuais: algoritmo desta Tese *versus* algoritmo de [43].

A série B é composta por 18 instâncias; para todas conseguiu-se provar otimalidade. Quanto à série C, as únicas instâncias para as quais não se conseguiu provar otimalidade foram a c10 e a c20, obtendo-se um salto de dualidade médio de 0.31%. Das 20 instâncias da série D apenas a d20 não foi testada, devido ao tamanho; provou-se otimalidade para 63.16% dos problemas enquanto em [43] prova-se otimalidade para todas as 20 instâncias. As instâncias da série E são muito grandes (repare que para utilizar a formulação (38) ainda é necessário dobrar o número de arestas) e possuem, muitas delas, um grande número de vértices terminais. Para esta série apenas 14 instâncias foram testadas, provando-se otimalidade para 57.14% delas, enquanto em [43] prova-se otimalidade para 19 das 20 instâncias, dentro de um tempo máximo de 10000 segundos. Vale ressaltar, porém, que em [43] o pré-processamento para diminuição do tamanho do grafo reduz em até 97% o número de arestas, em até 96% o número de vértices e em até 93% o número de vértices terminais. Pode-se concluir que para essas instâncias, definidas sobre grafos randômicos esparsos com custos euclidianos, o pré-processamento possui um papel *crucial*.

A série R foi de resolução bastante fácil para ambos algoritmos.

Na série ALUE conseguiu-se provar otimalidade para 30.77% das instâncias e melhorar o salto de dualidade em 7.69%, enquanto em [43] prova-se otimalidade para 53.85% das instâncias e o salto de dualidade é melhor em 38.46%. Isto significa que

o algoritmo apresentado em [43] foi melhor em 61.54% das instâncias. Sem dúvida essa foi a pior performance obtida pelo algoritmo desenvolvido neste trabalho. O pré-processamento de [43] foi capaz de reduzir em até 25% o número de arestas e em até 20% o número de vértices para esta série de problemas reais provenientes de circuitos eletrônicos VLSI. O número de vértices terminais é inalterado.

Na série GAP conseguiu-se provar otimalidade para 84.61% das instâncias enquanto em [43] prova-se otimalidade para todas elas. Isto significa que o algoritmo apresentado em [43] foi melhor em 15.39% das instâncias. O pré-processamento de [43] foi capaz de reduzir em até 20% o número de arestas e em até 25% o número de vértices para esta série. O número de vértices terminais permanece praticamente inalterado.

Na série TAQ conseguiu-se provar otimalidade para 71.43% das instâncias e melhorar o salto de dualidade em 7.14%, enquanto em [43] prova-se otimalidade para os mesmos 71.43% das instâncias e o salto de dualidade é melhor em 21.43%. Isto significa que o algoritmo apresentado em [43] foi melhor em 14.29% das instâncias. O pré-processamento de [43] foi capaz de reduzir em até 15% tanto o número de arestas quanto o número de vértices para esta série. O número de vértices terminais permanece praticamente inalterado.

Na série ALUT conseguiu-se provar otimalidade para 42.86% das instâncias e melhorar o salto de dualidade em também 42.86% delas, enquanto em [43] prova-se otimalidade para os mesmos 42.86% das instâncias e o salto de dualidade é melhor em apenas 14.28%. Isto significa que o algoritmo apresentado nesta Tese foi melhor em 28.58% das instâncias. O pré-processamento de [43] foi capaz de reduzir em menos de 10% o número de arestas e em menos de 15% o número de vértices para esta série. O número de vértices terminais permanece praticamente inalterado.

Na série DMXA conseguiu-se provar otimalidade para todas as instâncias com exceção de uma, para a qual conseguiu-se melhorar o salto de dualidade. Em [43] também prova-se otimalidade para aquelas mesmas instâncias. Isto significa que o algoritmo aqui desenvolvido conseguiu melhorar o salto de dualidade na única instância que não havia ainda sido resolvida: uma melhora de 100%. O pré-processamento de [43] foi capaz de reduzir em menos de 5% o número de arestas e em menos de 10% o número de vértices para esta série. O número de vértices terminais é inalterado.

Na série MSM conseguiu-se provar otimalidade para 90% das instâncias. Das 5 instâncias para as quais não se tinha provado otimalidade em [43], em apenas 1 o salto de dualidade deles é menor, e o algoritmo desenvolvido neste trabalho conseguiu melhorar o salto de dualidade em 2 das 5 instâncias e *provar otimalidade* para outras 2, a saber, msm2601 e msm4312. No total, para esta classe de instâncias, a melhora introduzida pelo algoritmo desenvolvido nesta Tese foi de 80%. O pré-processamento de [43] foi capaz de reduzir em menos de 5% o número de arestas e em menos de 10%

o número de vértices para esta série. O número de vértices terminais é inalterado.

Foi na série DIW que o algoritmo desenvolvido nesta Tese apresentou a melhor performance: das 7 instâncias para as quais não se tinha provado otimalidade em [43], melhorou-se o salto de dualidade em 5 delas e *provou-se otimalidade* para as outras 2, a saber, diw0795 e diw0801. Ou seja, a melhora introduzida pelo algoritmo aqui desenvolvido foi de 100%. O pré-processamento de [43] foi capaz de reduzir em menos de 5% tanto o número de arestas quanto o número de vértices para esta série. O número de vértices terminais é inalterado.

Conclusão. Os resultados mostram que para as séries de instâncias B, C, D e E, definidas sobre grafos randômicos esparsos com custos euclidianos, o pré-processamento do tamanho do grafo é *crucial* para a resolução de instâncias muito grandes. O pré-processamento torna-se mais relevante conforme aumenta o número de vértices terminais.

Para as instâncias definidas sobre reticulados derivadas de circuitos eletrônicos VLSI reais, pode-se observar que o pré-processamento já não consegue reduzir tanto o tamanho do grafo. Além disso, conforme o efeito do pré-processamento decresce, a performance do algoritmo desenvolvido nesta Tese cresce em relação ao algoritmo proposto em [43], o melhor até hoje conhecido para essa classe de problemas difíceis.

Vale ressaltar que o algoritmo desenvolvido neste trabalho não utiliza nenhum tipo de pré-processamento do tamanho do grafo e nenhum método enumerativo para encontrar soluções inteiras, utilizando apenas o limite inferior proveniente da resolução do problema dual e o limte superior fornecido pelas heurísticas primais.

Das 32 instâncias derivadas de circuitos eletrônicos VLSI reais, para as quais a otimalidade ainda não havia sido provada, *foi possível, com o algoritmo proposto nesta Tese, provar otimalidade para 4 instâncias, a saber msm2601, msm4312, diw0795 e diw0801, e melhorar o salto de dualidade em 14 outras instâncias; gerando, portanto, uma melhora de mais de* 53%. Pode-se concluir então que a performance obtida pelo algoritmo desenvolvido nesta Tese foi bastante satisfatória.

Nos resultados completos apresentados nos Apêndices 1, 2 e 3, pode-se ver que a heurística de Takahashi & Matsuyama com custos volumétricos foi a que gerou melhores limites superiores na maior parte das instâncias testadas. □

Apêndices

2.5 Comparação dos Algoritmos do Volume e do Volume Revisado

Nesta subseção atesta-se a boa performance do algoritmo do volume revisado, comparando-se RVA, RVA com os *testes de parada sérios split* (35) e VA aplicados a algumas instâncias da *SteinLib* disponíveis na literatura em [43] e via *ftp* anônimo no endereço eletrônico: `ftp://ftp.zib.de/pub/mp-testdata/SteinLib/`.

O código foi implementado em C++ em conjunto com o Dr. Francisco Barahona, e o computador utilizado foi um Pentium II 400MHz.

Foram testadas as 46 instâncias de *Soukup and Chow*, definidas sobre reticulados sem buracos; e as 21 instâncias reais de circuitos eletrônicos VLSI chamadas de *DIW*, definidas sobre reticulados com buracos (obstáculos).

Para simplificar a exibição dos resultados, as instâncias foram classificadas em quatro categorias, de acordo com a dificuldade de resolução:

- *Muito Fáceis* (MF): 35 instâncias *Soukup and Chow* com $|V| \leq 100$;
- *Fáceis* (F): as restantes 11 instâncias *Soukup and Chow* com $|V| > 100$ e 8 instâncias *DIW* com $|V| \leq 1000$ e $|T| \approx 5\%|V|$;
- *Médias* (M): 6 instâncias *DIW* com $1000 < |V| \leq 3000$, $|E_d| \leq 2|V|$ e $|T| \approx 2\%|V|$;
- *Difíceis* (D): as restantes 7 instâncias *DIW* com $|V| > 3000$, $|E_d| > 2|V|$ e $|T| \approx 1\%|V|$.

Os parâmetros de inicialização comuns a VA já foram descritos em § 2.2.1. Para o parâmetro de *tolerância séria* τ_s, utilizado no ([***Teste Sério***]) (34) do PASSO 4 de RVA, tem-se os seguintes parâmetros:

- $\tau_s = 0.1$, para os problemas classificados como MF e F;
- $\tau_s = 0.01$, para os problemas classificados como M e D.

Em vez de apresentar-se os resultados para cada uma das 67 instâncias, exibe-se performances médias dos algoritmos VA, RVA e RVA com os *testes de parada sérios-split* (35):

- a Tabela 2 mostra a percentagem média de passos nulos e sérios efetuados pelos algoritmos;

- a Tabela 3 mostra a percentagem média dos saltos de dualidade e os tempos médios de CPU.

A Tabela 2 abaixo, mostra que VA e RVA possuem comportamentos similares. Também mostra que RVA parece ser ligeiramente melhor ajustado para problemas mais difíceis, especialmente quando os *Testes-de-Parada-Sérios-Split* (35) são utilizados.

	Passos-sérios (%)			Passos-nulos (%)			Maior % de
Grupo	VA	RVA	RVA_{split}	VA	RVA	RVA_{split}	passos-sérios
MF	20.1	18.2	18.4	79.9	81.8	81.6	VA
F	17.8	16.6	17.1	82.2	83.4	82.9	VA
M	9.0	9.7	10.9	91	90.3	89.1	RVA_{split}
D	7.2	8.0	9.0	92.8	92.0	91.0	RVA_{split}

Tabela 2: Percentagem média de passos nulos e sérios.

A Tabela 3 abaixo, confirma que VA e RVA possuem comportamento similar, com RVA_{split} sendo mais eficiente para problemas mais difíceis.

	Salto de Dualidade(%)			Tempo Total de CPU(s)			Melhor
Grupo	VA	RVA	RVA_{split}	VA	RVA	RVA_{split}	Algoritmo
MF	0.0	0.0	0.0	0.3	0.5	0.7	VA
F	0.0	0.0	0.0	45.0	57.5	60.5	VA
M	0.0	0.0	0.0	4900.0	4750.0	4700.0	RVA_{split}
D	6.0	5.8	5.6	8100.0	7700.0	7500.0	RVA_{split}

Tabela 3: Percentagem média de salto de dualidade e tempo médio de CPU.

Observação 2.2 Conclui-se nesta seção § 2.5 que o algoritmo RVA_{split}, além ser suportado do ponto de vista teórico por uma análise de convergência, é o que apresenta performances numéricas mais robustas.

Observação 2.3 Uma pergunta óbvia a essa altura é sobre o porquê de não se ter utilizado RVA com os testes de parada sérios-split para a resolução de *todos* os problemas testados. Como pode-se imaginar, os resultados de convergência são mais difíceis e por conseguinte demoram muito mais para ficar prontos. Enquanto isso os testes computacionais foram efetuados com VA. Devido ao tempo limitado para a execução do doutorado não foi possível realizar os testes com RVA_{split} para as demais instâncias, optou-se por testá-lo naquelas classes de instâncias onde VA já havia demonstrado uma boa performance. Pretende-se continuar os testes com RVA_{split} como uma extensão natural deste trabalho.

3 Conclusões e Extensões

A aparição do algoritmo do volume foi, sem dúvida, muito importante para a resolução de problemas de otimização combinatória, num contexto de relaxação Lagrangeana.

O algoritmo desenvolvido neste trabalho para o problema de Steiner em grafos (SPG) foi mais uma prova disso, o que soma o SPG à lista dos problemas difíceis de otimização combinatória que já haviam sido abordados com sucesso com o algoritmo do volume [5, 6, 7, 8], tais como: *set partitioning, set covering, max-cut* e *facility location.*

A qualidade dos resultados obtidos para o SPG com o algoritmo desenvolvido nesta Tese foi comparável à qualidade daqueles resultados apresentados em [43], até hoje os melhores conhecidos na literatura. É importante ressaltar que os resultados apresentados em § 2.4 e nos Apêndices foram obtidos sem nenhum tipo de pré-processamento para reduzir o tamanho das instâncias originais e sem nunca recorrer a métodos de enumeração implícita, tais como *branch-and-bound*, enquanto que em [43] são utilizados.

A versão revisada do algoritmo do Volume (RVA) aqui introduzida, embora diferindo ligeiramente do algoritmo do volume original (VA), representou uma contribuição importante pela apresentação de uma prova de convergência e de um limite *a posteriori* para o erro de aproximação das soluções estimadas produzidas. Ao longo da prova de convergência do algoritmo do volume revisado, mostrou-se que o algoritmo do volume na verdade não se encaixava na classe dos métodos de subgradientes, como pensavam originalmente seus autores, sendo melhor classificado como um método de extragradientes. Os testes computacionais feitos com VA e RVA em § 2.5 mostraram que estes possuem comportamentos similares. Também mostraram que RVA parece ser ligeiramente melhor ajustado para problemas mais difíceis, especialmente quando os testes de parada sérios-split (35) são utilizados. Por isso uma extensão natural deste trabalho consiste em aplicar $\text{RVA}_{\text{split}}$ a todos os problemas da biblioteca *SteinLib.*

Uma interessante conseqüência da intenção inicial de provar a convergência de VA foi o desenvolvimento de um novo padrão algorítmico para problemas de otimização não-suaves (Non Smooth Optimization, NSO) no qual um número infinito de passos-sérios é gerado. Uma vez que sua principal característica consiste em atualizar os multiplicadores de Lagrange através de um deslocamento de ε-extragradiente, chamou-se este método de ε-ESG.

Várias são as extensões naturais resultantes deste trabalho. O novo padrão algorítmico para problemas de otimização não suaves pode ser testado para outros problemas não-diferenciáveis que não provenientes de relaxações Lagrangeanas de problemas otimização combinatória.

A aplicação do algoritmo do volume ao SPG mostrou que este algoritmo permite lidar computacionalmente com a formulação multi-fluxo proposta simultaneamente por Claus & Maculan [23] e Wong [76]. Essa formulação pode ser utilizada *mutatis mutandis* para diversos problemas de desenho de redes (*network design problems*). Em particular, um algoritmo para a resolução do *prize collecting Steiner tree problem, PCSTP* baseado numa formulação multi-fluxos parecida à utilizada para o SPG, já está sendo desenvolvido, também em conjunto com o Dr. Francisco Barahona. O PCSTP difere do SPG no tocante à existência de penalidades não negativas para os vértices do grafo.

Finalmente, novas heurísticas baseadas nas soluções primais estimadas pelo Algoritmo do Volume podem ser desenvolvidas para outros problemas de otimização combinatória. De fato, uma heurística desse tipo foi utilizada com sucesso recentemente para o Problema Quadrático da Mochila em [56].

Apêndice 1

Problemas das Séries B, C, D e E

A Tabela 3 é relativa aos problemas-teste das séries B e C, introduzidos por J. Beasley [11]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	*LB*	*UB*	*Gap*(%)	$CPU_{\text{Tempo}}(s)$	*MSTV*	*MMSTV*	*T&MV*	*PC*
b01	50	63	9	81.563	82	0.000	0.350	82	83	85	≈
b02	50	63	13	82.766	83	0.000	0.440	83	84	86	≈
b03	50	63	25	137.585	138	0.000	1.100	139	138	138	≈
b04	50	100	9	59.000	59	0.000	0.430	59	59	62	≈
b05	50	100	13	60.977	61	0.000	0.890	61	62	62	≈
b06	50	100	25	121.984	122	0.000	2.740	122	124	128	≈
b07	75	94	13	110.406	111	0.000	0.650	111	111	111	≈
b08	75	94	19	103.139	104	0.000	1.750	104	108	104	≈
b09	75	94	38	219.397	220	0.000	4.160	226	220	222	≈
b10	75	150	13	85.568	86	0.000	0.650	91	86	90	≈
b11	75	150	19	87.997	88	0.000	1.870	88	90	90	≈
b12	75	150	38	173.913	174	0.000	8.200	174	175	177	≈
b13	100	125	17	165.000	165	0.000	2.140	165	172	178	≈
b14	100	125	25	235.000	235	0.000	2.980	235	238	241	≈
b15	100	125	50	317.639	318	0.000	11.660	318	323	322	≈
b16	100	200	17	126.997	127	0.000	3.460	127	142	136	≈
b17	100	200	25	130.111	131	0.000	4.340	131	132	132	≈
b18	100	200	50	217.992	218	0.000	14.490	218	219	223	≈
c01	500	625	5	84.272	85	0.000	1.790	85	85	85	≈
c02	500	625	10	143.528	144	0.000	6.720	144	175	144	≈
c03	500	625	83	753.039	754	0.000	192.950	754	758	769	≈
c04	500	625	125	1078.020	1079	0.000	449.920	1079	1079	1101	≈
c05	500	625	250	1578.980	1579	0.000	5379.220	1579	1586	1587	≈
c06	500	1000	5	54.055	55	0.000	2.630	55	74	55	≈
c07	500	1000	10	101.809	102	0.000	7.170	107	127	102	≈
c08	500	1000	83	508.900	509	0.000	216.420	516	509	521	≈
c09	500	1000	125	706.987	707	0.000	4058.820	707	709	708	≈
c10	500	1000	250	1092.990	1094	0.092	10501.200	1094	1094	1094	$*^{\dagger}$
c11	500	2500	5	31.995	32	0.000	11.320	32	33	34	≈
c12	500	2500	10	45.892	46	0.000	26.240	46	48	48	≈
c13	500	2500	83	257.351	258	0.000	1700.650	259	261	258	≈
c14	500	2500	125	322.039	323	0.000	664.410	333	323	334	≈
c15	500	2500	250	555.473	556	0.000	3506.140	562	557	556	≈
c16	500	12500	5	10.388	11	0.000	17.820	20	12	11	≈
c17	500	12500	10	17.959	18	0.000	100.270	18	20	19	≈
c18	500	12500	83	112.063	113	0.000	5043.900	114	115	113	≈
c19	500	12500	125	145.031	146	0.000	3325.490	146	148	159	≈
c20	500	12500	250	265.566	267	0.540	10425.400	267	267	268	$*^{\dagger}$

Tabela 4: Séries B e C: grafos randômicos esparsos com custos euclidianos.

A Tabela 3 é relativa aos problemas-teste das séries D e E, introduzidos por J. Beasley [11]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	LB	UB	$Gap(\%)$	$CPU_{\text{Tempo}}(s)$	$MSTV$	$MMSTV$	$T\&MV$	PC
d01	1000	1250	5	106.000	106	0.000	8.740	106	108	107	$\sim$
d02	1000	1250	10	219.413	220	0.000	14.940	229	220	220	$\sim$
d03	1000	1250	167	1564.990	1565	0.000	4840.800	1565	1567	1565	$\sim$
d04	1000	1250	250	1934.530	1935	0.000	3370.140	1937	1940	1935	$\sim$
d05	1000	1250	500	3237.340	3252	0.453	10424.300	3266	3260	3252	$*^{-1}$
d06	1000	2000	5	66.634	67	0.000	8.120	67	84	70	$\sim$
d07	1000	2000	10	102.458	103	0.000	19.900	103	105	103	$\sim$
d08	1000	2000	167	1071.990	1074	0.188	10427.800	1076	1079	1074	$*^{-1}$
d09	1000	2000	250	1447.080	1449	0.133	10509.800	1454	1454	1449	$*^{-1}$
d10	1000	2000	500	2107.840	2113	0.245	10505.100	2129	2113	2114	$*^{-1}$
d11	1000	5000	5	29.000	29	0.000	24.140	29	31	30	$\sim$
d12	1000	5000	10	41.114	42	0.000	36.220	42	47	42	$\sim$
d13	1000	5000	167	499.960	500	0.000	10509.800	504	504	500	$\sim$
d14	1000	5000	250	665.168	667	0.275	10501.700	677	669	667	$*^{-1}$
d15	1000	5000	500	1110.180	1116	0.524	10504.800	1136	1122	1116	$*^{-1}$
d16	1000	25000	5	12.029	13	0.000	42.430	20	14	13	$\sim$
d17	1000	25000	10	22.121	23	0.000	165.740	31	25	23	$\sim$
d18	1000	25000	167	222.188	223	0.000	10502.900	228	223	246	$\sim$
d19	1000	25000	250	307.578	314	2.088	10500.300	329	314	315	$*^{-1}$
e01	2500	3125	5	110.937	111	0.000	19.310	111	136	115	$\sim$
e02	2500	3125	10	214.000	214	0.000	53.400	214	218	227	$\sim$
e03	2500	3125	417	4002.990	4031	0.700	10508.400	4049	4045	4031	$*^{-1}$
e04	2500	3125	625	5086.800	5123	0.712	10508.400	5240	5217	5123	$*^{-1}$
e05	2500	3125	1250	8047.540	8164	1.447	10503.800	8527	8164	8212	$*^{-1}$
e06	2500	5000	5	73.000	73	0.000	20.220	73	85	78	$\sim$
e07	2500	5000	10	145.000	145	0.000	86.360	145	145	149	$\sim$
e08	2500	5000	417	2630.210	2658	1.057	10502.000	2739	2760	2658	$*^{-1}$
e09	2500	5000	625	3594.350	3612	0.491	10501.000	3680	3612	3737	$*^{-1}$
e11	2500	12500	5	33.499	34	0.000	35.990	46	34	38	$\sim$
e12	2500	12500	10	67.000	67	0.000	7066.980	67	72	68	$\sim$
e13	2500	12500	417	1270.230	1295	1.950	10505.000	1443	1295	1308	$*^{-1}$
e16	2500	62500	5	14.993	15	0.000	263.270	15	18	17	$\sim$
e17	2500	62500	10	24.879	25	0.000	760.210	25	26	26	$\sim$

Tabela 5: Séries D e E: grafos randômicos esparsos com custos euclidianos.

Apêndice 2

Problemas da Série R

A Tabela 6 é relativa aos primeiros 23 problemas-teste da série R, introduzidos por J. Soukup e W. F. Chow [66]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	*LB*	*UB*	*Gap*(%)	$CPU_{\text{Tempo}}(s)$	MSTV	MMSTV	T&MV	*PC*
r01	15	22	5	186.123	187	0.000	0.060	187	189	187	∼
r02	12	17	6	163.625	164	0.000	0.050	168	164	168	∼
r03	28	45	7	235.970	236	0.000	0.240	236	236	237	∼
r04	64	112	8	253.061	254	0.000	0.800	255	264	254	∼
r05	12	17	6	225.425	226	0.000	0.100	229	229	226	∼
r06	24	38	12	241.161	242	0.000	0.290	242	242	245	∼
r07	30	49	12	247.420	248	0.000	0.450	251	248	254	∼
r08	24	37	12	235.924	236	0.000	0.410	239	236	242	∼
r09	15	22	7	164.000	164	0.000	0.130	164	172	172	∼
r10	36	60	6	176.261	177	0.000	0.230	192	184	177	∼
r11	30	49	6	143.275	144	0.000	0.180	144	144	147	∼
r12	27	42	9	179.143	180	0.000	0.340	180	180	180	∼
r13	42	71	9	149.100	150	0.000	0.500	150	150	150	∼
r14	36	60	12	259.279	260	0.000	0.670	260	260	260	∼
r15	100	180	14	147.100	148	0.000	2.580	150	148	148	∼
r16	9	12	3	159.436	160	0.000	0.040	160	160	160	∼
r17	48	82	10	199.127	200	0.000	0.600	200	200	200	∼
r18	182	337	62	403.896	404	0.000	246.260	404	405	408	∼
r19	168	310	14	187.954	188	0.000	5.010	190	188	190	∼
r20	6	7	3	111.573	112	0.000	0.000	142	142	112	∼
r21	15	22	5	192.000	192	0.000	0.050	196	196	192	∼
r22	16	24	4	62.164	63	0.000	0.080	63	63	63	∼
r23	16	24	4	64.045	65	0.000	0.040	68	65	65	∼

Tabela 6: Série R: reticulados sem buracos.

A Tabela 7 é relativa aos últimos 23 problemas-teste da série R, introduzidos por J. Soukup e W. F. Chow [66]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	*LB*	*UB*	*Gap*(%)	$CPU_{\text{Tempo}}(s)$	MSTV	MMSTV	T&MV	*PC*
r24	16	24	4	29.407	30	0.000	0.060	30	30	30	≃
r25	9	12	3	22.880	23	0.000	0.020	23	23	23	≃
r26	9	12	3	14.572	15	0.000	0.010	17	17	15	≃
r27	16	24	4	132.052	133	0.000	0.100	133	133	133	≃
r28	12	17	4	23.446	24	0.000	0.060	24	24	24	≃
r29	9	12	3	199.387	200	0.000	0.050	200	200	200	≃
r30	28	45	12	109.210	110	0.000	0.540	110	110	110	≃
r31	130	237	14	258.084	259	0.000	4.660	260	259	259	≃
r32	210	391	19	313.000	313	0.000	41.280	313	314	317	≃
r33	132	241	18	267.024	268	0.000	6.030	268	270	269	≃
r34	272	511	19	240.991	241	0.000	26.430	251	246	241	≃
r35	240	449	18	150.028	151	0.000	12.110	154	152	151	≃
r36	6	7	4	90.000	90	0.000	0.000	98	90	90	≃
r37	49	84	8	89.103	90	0.000	0.330	91	92	90	≃
r38	100	180	14	165.217	166	0.000	2.430	183	166	181	≃
r39	100	180	14	165.623	166	0.000	2.170	176	166	181	≃
r40	64	112	10	154.210	155	0.000	0.850	175	158	155	≃
r41	144	263	20	223.988	224	0.000	9.970	230	226	224	≃
r42	81	144	15	152.185	153	0.000	1.980	157	159	153	≃
r43	195	362	16	254.926	255	0.000	8.840	255	256	256	≃
r44	196	364	17	251.996	252	0.000	14.700	256	252	253	≃
r45	270	507	19	220.000	220	0.000	147.370	220	224	224	≃
r46	16	24	16	149.137	150	0.000	0.400	150	150	150	≃

Tabela 7: Série R: reticulados sem buracos.

Apêndice 3

Problemas Derivados de Circuitos VLSI Reais

A Tabela 8 é relativa aos problemas reais das séries ALUE e TAQ, introduzidos por T. Koch e A. Martin [43]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	*LB*	*UB*	*Gap*(%)	$CPU_{\text{Tempo}}(s)$	*MSTV*	*MMSTV*	*T&MV*	*PC*
alue2087	1244	1971	34	1049.000	1049	0.000	1433.910	1054	1063	1049	$\sim$
alue2105	1220	1858	34	1031.002	1032	0.000	734.340	1032	1032	1032	$\sim$
alue3146	3626	5869	64	2004.110	2242	11.870	10520.200	2380	3346	2242	$-$
alue5067	3524	5560	68	2586.000	2586	0.000	10519.800	2586	3012	2623	$\sim$
alue5345	5179	8165	68	3238.840	3566	10.101	10550.500	3781	4463	3566	$-$
alue5623	4472	6938	68	3204.560	3480	8.595	10546.300	3839	3913	3480	$-$
alue5901	11543	18429	68	3322.830	4011	20.710	10761.600	5221	4831	4011	$-$
alue6179	3372	5213	67	2386.330	2457	2.961	10531.200	2567	2707	2457	$*^{-}$
alue6457	3932	6137	68	2997.970	3067	2.250	10554.200	3398	3450	3067	$-$
alue6735	4119	6696	68	2651.440	2704	1.982	10547.200	2776	2887	2704	$*^{-}$
alue6951	2818	4419	67	2381.470	2403	0.904	10516.200	2465	2445	2403	$*^{-}$
alue7066	6405	10454	16	2194.760	2275	3.655	10520.300	2540	2340	2275	$+$
alue7229	940	1474	34	823.202	824	0.000	83.460	827	899	824	$\sim$
taq0014	6466	11046	128	4295.260	5459	27.094	10457.700	6994	6681	5459	$-$
taq0023	572	963	11	621.000	621	0.000	207.370	621	623	623	$\sim$
taq0365	4186	7074	22	1911.110	1914	0.151	10506.500	1981	1963	1914	$+$
taq0377	6836	11715	136	5403.270	6551	21.241	10489.800	8242	8130	6551	$-$
taq0431	1128	1905	13	896.002	897	0.000	382.910	897	946	897	$\sim$
taq0631	609	932	10	581.000	581	0.000	85.950	588	588	581	$\sim$
taq0739	837	1438	16	847.500	848	0.000	8537.940	848	885	848	$\sim$
taq0741	712	1217	16	846.998	847	0.000	699.930	847	856	855	$\sim$
taq0751	1051	1791	16	938.420	939	0.000	10501.200	939	949	939	$\sim$
taq0891	331	560	10	318.124	319	0.000	10.300	331	321	319	$\sim$
taq0903	6163	10490	130	4284.170	5195	21.260	10486.800	6809	6783	5195	$-$
taq0910	310	514	17	369.165	370	0.000	8.690	427	407	370	$\sim$
taq0920	122	194	17	209.077	210	0.000	2.100	217	212	210	$\sim$
taq0978	777	1239	10	565.175	566	0.000	48.200	649	613	566	$\sim$

Tabela 8: Séries ALUE e TAQ: reticulados com buracos (circuitos VLSI reais).

A Tabela 9 é relativa aos problemas reais das séries GAP e ALUT, introduzidos por T. Koch e A. Martin [43]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	*LB*	*UB*	*Gap*(%)	$CPU_{\text{Tempo}}(s)$	*MSTV*	*MMSTV*	*T&MV*	*PC*
gap1307	342	552	17	548.896	549	0.000	15.180	549	551	551	$\sim$
gap1413	541	906	10	456.033	457	0.000	14.070	464	478	457	$\sim$
gap1500	220	374	17	253.108	254	0.000	10.490	301	266	254	$\sim$
gap1810	429	702	17	481.932	482	0.000	19.630	496	482	489	$\sim$
gap1904	735	1256	21	762.960	763	0.000	48.900	763	763	785	$\sim$
gap2007	2039	3548	17	1099.000	1104	0.455	10001.600	1104	1114	1104	$*^{-}$
gap2119	1724	2975	29	1243.010	1244	0.000	3073.950	1244	1244	1244	$\sim$
gap2740	1196	2084	14	744.997	745	0.000	1121.180	758	775	745	$\sim$
gap2800	386	653	12	385.477	386	0.000	21.170	401	400	386	$\sim$
gap2975	179	293	10	244.920	245	0.000	2.620	245	245	245	$\sim$
gap3036	346	583	13	456.985	457	0.000	15.740	457	457	465	$\sim$
gap3100	921	1558	11	639.999	640	0.000	605.100	656	649	640	$\sim$
gap3128	10393	18043	104	3453.180	4407	27.622	10445.700	5462	5724	4407	$*^{-}$
alut0787	1160	2089	34	981.005	982	0.000	1299.150	982	1064	982	$\sim$
alut0805	966	1666	34	957.001	958	0.000	2982.070	958	958	963	$\sim$
alut1181	3041	5693	64	2334.090	2377	1.838	10420.500	2612	2993	2377	$+$
alut2010	6104	11011	68	3190.870	3355	5.144	10472.500	3912	4390	3355	$-$
alut2288	9070	16595	68	3541.740	3892	9.889	10409.700	5825	4984	3892	$+$
alut2566	5021	9055	68	2952.420	3117	7.341	10449.900	3326	4559	3117	$+$
alut2764	387	626	34	639.995	640	0.000	53.450	642	663	640	$\sim$

Tabela 9: Séries GAP e ALUT: reticulados com buracos (circuitos VLSI reais).

A Tabela 10 é relativa aos problemas reais das séries DMXA e DIW, introduzidos por T. Koch e A. Martin [43]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	*LB*	*UB*	*Gap*(%)	$CPU_{\text{Tempo}}(s)$	*MSTV*	*MMSTV*	*T&MV*	*PC*
dmxa0296	233	386	12	343.062	344	0.000	3.720	416	366	344	$\sim$
dmxa0368	2050	3676	18	1016.990	1017	0.000	4696.280	1017	1031	1021	$\sim$
dmxa0454	1848	3286	16	913.997	914	0.000	1535.710	941	939	914	$\sim$
dmxa0628	169	280	10	274.959	275	0.000	3.710	275	277	277	$\sim$
dmxa0734	663	1154	11	506.000	506	0.000	343.070	507	523	506	$\sim$
dmxa0848	499	861	16	593.997	594	0.000	40.180	599	594	599	$\sim$
dmxa0903	632	1087	10	580.000	580	0.000	126.850	580	595	585	$\sim$
dmxa1010	3983	7108	23	1436.290	1493	3.948	10503.600	1650	1619	1493	+
dmxa1109	343	559	17	453.954	454	0.000	15.310	454	454	462	$\sim$
dmxa1200	770	1383	21	749.045	750	0.000	94.750	754	750	750	$\sim$
dmxa1304	298	503	10	310.999	311	0.000	7.260	311	311	312	$\sim$
dmxa1516	720	1269	11	507.844	508	0.000	24.010	508	508	513	$\sim$
dmxa1721	1005	1731	18	779.998	780	0.000	132.400	780	781	781	$\sim$
dmxa1801	2333	4137	17	1364.002	1365	0.000	3226.410	1390	1365	1365	$\sim$
diw0234	5349	10086	25	1995.970	1996	0.000	10512.800	2192	2192	1996	$\sim$
diw0250	353	608	11	349.099	350	0.000	5.590	431	350	350	$\sim$
diw0260	539	985	12	467.152	468	0.000	54.890	751	475	468	$\sim$
diw0313	468	822	14	396.063	397	0.000	14.390	409	399	397	$\sim$
diw0393	212	381	11	301.097	302	0.000	3.960	302	302	302	$\sim$
diw0445	1804	3311	33	1362.030	1363	0.000	7268.660	1414	1458	1363	$\sim$
diw0459	3636	6789	25	1361.020	1362	0.000	7013.390	1808	1554	1362	$\sim$
diw0460	339	579	13	344.002	345	0.000	5.040	376	388	345	$\sim$
diw0473	2213	4135	25	1097.030	1098	0.000	3891.350	1254	1098	1100	$\sim$
diw0487	2414	4386	25	1423.970	1424	0.000	8630.050	1434	1438	1424	$\sim$
diw0495	938	1655	10	615.223	616	0.000	92.040	1036	616	634	$\sim$
diw0513	918	1684	10	604.000	604	0.000	140.710	609	702	604	$\sim$
diw0523	1080	2015	10	560.112	561	0.000	58.590	647	587	561	$\sim$
diw0540	286	465	10	373.088	374	0.000	3.370	397	431	374	$\sim$
diw0559	3738	7013	18	1489.940	1570	5.373	10503.100	1778	1860	1570	+
diw0778	7231	13727	24	2101.110	2173	3.421	10420.300	2933	2570	2173	+
diw0779	11821	22516	50	3346.010	4598	37.417	10404.200	7349	6422	4598	+
diw0795	3221	5938	10	1549.140	1550	0.000	2604.880	1757	2072	1550	$*^{+}$
diw0801	3023	5575	10	1586.020	1587	0.000	2793.660	1602	1795	1587	$*^{+}$
diw0819	10553	20066	32	3038.360	3404	12.034	10454.900	5937	4581	3404	+
diw0820	11749	22384	37	3789.430	4240	11.890	10428.500	6420	6634	4240	+

Tabela 10: Séries DMXA e DIW: reticulados com buracos (circuitos VLSI reais).

A Tabela 11 é relativa aos primeiros 15 problemas reais da série MSM, introduzidos por T. Koch e A. Martin [43]:

Nome	$\|V\|$	$\|E\|$	$\|T\|$	*LB*	*UB*	*Gap*(%)	$CPU_{\text{Tempo}}(s)$	*MSTV*	*MMSTV*	*T&MV*	*PC*
msm0580	338	541	11	466.999	467	0.000	8.940	467	467	480	$\sim$
msm0654	1290	2270	10	822.989	823	0.000	1153.420	833	833	823	$\sim$
msm0709	1442	2403	16	883.002	884	0.000	396.170	897	896	884	$\sim$
msm0920	752	1264	26	805.053	806	0.000	93.590	806	806	808	$\sim$
msm1008	402	695	11	494.000	494	0.000	169.930	498	498	494	$\sim$
msm1234	933	1632	13	550.000	550	0.000	733.080	550	563	563	$\sim$
msm1477	1199	2078	31	1067.010	1068	0.000	1108.140	1086	1089	1068	$\sim$
msm1707	278	478	11	563.037	564	0.000	4.860	564	564	564	$\sim$
msm1844	90	135	10	187.947	188	0.000	1.180	188	215	188	$\sim$
msm1931	875	1522	10	603.002	604	0.000	168.070	634	735	604	$\sim$
msm2000	898	1562	10	593.179	594	0.000	88.080	712	670	594	$\sim$
msm2152	2132	3702	37	1590.000	1590	0.000	6868.560	1603	1631	1590	$\sim$
msm2326	418	723	14	398.454	399	0.000	10.780	407	399	399	$\sim$
msm2492	4045	7094	12	1456.480	1459	0.173	10501.800	1525	1459	1459	$+$
msm2525	3031	5239	12	1289.980	1290	0.000	5523.380	1417	1310	1290	$\sim$
msm2601	2961	5100	16	1439.990	1440	0.000	6929.590	1450	1583	1440	$*^{+}$
msm2705	1359	2458	13	713.120	714	0.000	155.710	805	727	714	$\sim$
msm2802	1709	2963	18	925.014	926	0.000	2251.200	933	933	926	$\sim$
msm2846	3263	5783	89	3036.580	3153	3.834	10442.700	3384	3548	3153	$-$
msm3277	1704	2991	12	868.024	869	0.000	1755.490	930	942	869	$\sim$
msm3676	957	1554	10	606.001	607	0.000	82.250	627	622	607	$\sim$
msm3727	4640	8255	21	1375.020	1376	0.000	6279.820	1557	1409	1376	$\sim$
msm3829	4221	7255	12	1564.490	1593	1.822	10502.600	1973	1729	1593	$+$
msm4038	237	390	11	352.029	353	0.000	4.460	356	353	358	$\sim$
msm4114	402	690	16	392.292	393	0.000	9.440	429	393	393	$\sim$
msm4190	391	666	16	380.106	381	0.000	24.380	423	381	381	$\sim$
msm4224	191	302	11	310.001	311	0.000	3.100	311	311	311	$\sim$
msm4312	5181	8893	10	2015.980	2016	0.000	8943.140	2016	2095	2016	$*^{+}$
msm4414	317	476	11	407.141	408	0.000	11.910	437	408	408	$\sim$
msm4515	777	1358	13	630.000	630	0.000	231.700	640	640	630	$\sim$

Tabela 11: Série MSM: reticulados com buracos (circuitos VLSI reais).

Referências Bibliográficas

Referências Bibliográficas

[1] AGARWAL, P. K., SHING, M. T., "Algorithms for The Special Cases of Rectilinear Steiner Trees: Points on the Boundary of a Rectilinear Rectangle", *Networks*, v.20, pp. 453–485, 1990.

[2] ANEJA, Y. P., "An Integer Linear Programming Approach to the Steiner Problem in Graphs", *Networks*, v.10, pp. 167–178, 1980.

[3] BAHIENSE, L., MACULAN, N., SAGASTIZÁBAL, C., *On the Convergence of the Volume Algorithm*, Universidade Federal do Rio de Janeiro ES-519/99, Submetido para publicação em *Mathematical Programming* em Dezembro de 1999.

[4] BAHIENSE, L., BARAHONA, F., *Solving Steiner Tree Problems in Graphs with the Volume Algorithm*, T.J.Wtason Research Center, IBM, NY–USA, *Working Paper*, A ser submetido para publicação em *Journal of Combinatorial Optimization*, 1999.

[5] BARAHONA, F., ANBIL, R., *The Volume Algorithm: Producing Primal Solutions with a Subgradient Method*, IBM Research Report RC 21103, Aceito para publicação em *Mathematical Programming*, 1998.

[6] BARAHONA, F., ANBIL, R., *On Some Difficult Linear Programs Coming from Set Partitioning*, IBM Research Report RC 21410, 1999.

[7] BARAHONA, F., CHUDAK, F., *Solving Large Scale Uncapacitated Location Problems*, IBM Research Report RC 21515, 1999.

[8] BARAHONA, F., CHUDAK, F., *Near-optimal Solutions to Large Scale Facility Location Problems*, IBM Research Report RC 21606, 1999.

[9] BARAHONA, F., JENSEN, D., *Plant Location with Minimum Inventory*, IBM Research Report RC 20367, 1996.

[10] BEASLEY, J., "Lagrangean Relaxation". In: Reeves, C. R. (ed), *Modern Heuristic Techniques for Combinatorial Problems*, 1 ed., chapter 6, Oxford-Great Britain, John Wiley & Sons, 1993.

[11] BEASLEY, J., "An SST-based Algorithm for the Steiner Problem in Graphs", *Networks*, v. 19, pp. 1–16, 1989.

[12] BERMANN, P., RAMAIYER, V., "Improved Approximations for the Steiner Tree Problem", *Journal Algorithms*, v. 17, pp. 381–408, 1994.

[13] BERN, M. W., *Network Design Problems: Steiner Trees and Spanning k-Trees.* Ph.D. dissertion, Department of Computer Science, University of Berkeley, Berkeley, California, USA, 1987.

[14] BERN, M. W., "Faster Exact Algorithm for Steiner Trees in Planar Networks", *Networks*, v. 20, pp. 109–120, 1990.

[15] BERN, M. W., PLASSMANN, P., "The Steiner Problem with Edge Lenghts 1 and 2", *Information Processing Letters*, v. 32, pp. 171–176, 1989.

[16] BERTSEKAS, D. P., *Nonlinear Programming.* 1 ed. Massachusetts, Athena Scientific, 1995.

[17] BERTSIMAS, D., TSITSIKLIS, J. N., *Introduction to Linear Optimization.* 1 ed. Massachusetts, Athena Scientific, 1997.

[18] BONNANS, J. F., GILBERT, J. Ch., LEMARÉCHAL, C., SAGASTIZÁBAL, C., *Optimisation Numérique: aspects théoriques et pratiques.* 1 ed. Berlin, Springer Verlag, 1997.

[19] CAMPELLO, R. E., MACULAN, N., *Algoritmos e Heurísticas, Desenvolvimento e Avaliação de Performance.* 1ª ed. Rio de Janeiro, Brasil, Editora da Universidade Federal Fluminense, 1994.

[20] CHOPRA, S., GORRES, E. R., RAO, M. R., "Solving the Steiner Tree Problem in Graphs Using Branch-and-cut", *ORSA Journal on Computation*, v. 4, pp. 320–335, 1992.

[21] CHOPRA, S., RAO, M. R., "The STeiner Tree Problem I: Formulations, Compositions and Extension of Facets", *Mathematical Programming*, v. 64, pp. 209–229, 1994.

[22] CHOPRA, S., RAO, M. R., "The STeiner Tree Problem II: Properties and Classes of Facets", *Mathematical Programming*, v. 64, pp. 231–246, 1994.

[23] CLAUS, A., MACULAN, N., *Une Nouvelle Formulation du Problème de Steiner sur un Graphe*, Centre de Reserche sur les Transports, Université de Montréal, Technical Report 280, 1983.

[24] CORREA, R., LEMARÉCHAL, C., "Convergence of Some Algorithms for Convex Minimization", *Mathematical Programming*, v. 62, n. 2, pp. 261–275, 1993.

[25] COURANT, R., ROBBINS, H., *What is Matemathics?* 1 ed. New York, Oxford University Press, 1941.

[26] DANTZIG, G. B., WOLFE, P., "Decomposition Principle for Linear Programs", *Operations Research*, v. 8, pp. 101–111, 1960.

[27] DOWSLAND, K. A., "Hill-climbing, Simulated Annealing and the Steiner Tree Problem in Graphs", *Engeneering Optimization*, v. 17, pp. 91–107, 1991.

[28] DUIN, C. W., *Steiner Problem in Graphs: Approximation, Reduction, Variation.* Ph.D. dissertion, Institute of Actuarial Science & Economics, University of Amsterdam, Amsterdam, The Netherlands, 1994.

[29] DUIN, C. W., VOß, S., "Steiner Tree Heuristics: a Survey". In: *Operations Research Proceedings 1993*, pp. 485–496, Berlin, 1994.

[30] DUIN, C. W., VOß, S., "Efficient Path and Vertex Exchange in Steiner Tree Algorithms", *Networks*, v. 29, pp. 89–105, 1997.

[31] ESBENSEN, H., "Computing Near-optimal Solutions to the Steiner Tree Problem in a Graph Using a Genetic Algorithm", *Networks*, v. 26, pp. 173–185, 1995.

[32] EVERETT, H., "Generalized Lagrange Multipliers for Solving Problems of Optimum Allocation of Resources", *Operations Research*, v. 11, pp. 399–417, 1963.

[33] GANLEY, J. L., "Computing Optimal Rectilinear Steiner Trees: a Survey and Experimental Evaluation", *Discrete Applied Mathematics*, v. 90, pp. 161–171, 1999.

[34] GAREY, M. R., JOHNSON, D. S., "The Rectilinear Steiner Tree Problem is *NP*-complete", *SIAM Journal on Applied Mathematics*, v. 32, pp. 826–834, 1977.

[35] GOEMANS, M. X. , MYUNG, Y., S., "A Catolog of Steiner Tree Formulations", *Networks*, v. 23, pp. 19–28, 1993.

[36] HELD, M., WOLFE, P., CROWDER, H. P., "Validation of Subgradient Algorithm", *Mathematical Programming*, v. 6, pp. 62–88, 1974.

[37] HIRIART-URRUTY, J. B., LEMARÉCHAL, C., *Convex analysis and minimization algorithms.* 1 ed. Berlin, Springer Verlag, 1991.

[38] HWANG,F. K., RICHARDS, D. S., WINTER, P., *The Steiner Tree Problem*, 1 ed. Amsterdam, North Holland, 1992.

[39] KAPSALIS, A., RAYWARD-SMITH, V. J., SMITH, G. D., "Solving the Graphical Steiner Tree Problem Using Genetic Algorithms", *Journal of the Operational Research Society*, v. 44, pp. 397–406, 1993.

[40] KARP, R. M., "Reducibility Among Combinatorial Problems", In: Miller, R. E., Thatcher, J. W. (eds), *Complexity of Computer Computations*, 1 ed., pp. 85–103, USA, Plenum Press, 1972.

[41] KELLEY, J. E., "The Cutting Plane Method for Solving Convex Programs", *Journal of the Societ of Industrial Applied Mathematics*, v. 8, pp. 703–712, 1960.

[42] KIWIEL, K. C., "Proximity Control in Bundle Methods for Convex Nondifferentiable Minimization", *Mathematical Programming*, v. 46, pp. 105–122, 1990.

[43] KOCH, T., MARTIN, A., "Solving STeiner Tree Problems in Graphs to Optimality", *Networks*, v. 32, pp. 207–232, 1998.

[44] KORPELEVICH, G. M., "The Extragradient Method for Finding Saddle Points and Other Problems", *Matecon*, v. 12, pp. 747–756, 1976.

[45] KUHN, H. W., "On a Pair of Dual Nonlinear Programs". In: Abadie, J. (ed), *Nonlinear Programming*, 1 ed., Amsterdam, North Holland, 1967.

[46] LEMARÉCHAL, C., "An Extension of Davidon Methods to Nondifferentiable Problems", *Mathematical Programming Study*, v. 3, pp. 95–109, 1975.

[47] LUCENA, A. "Steiner Problem in Graphs: Lagrangean Relaxation and Cutting-planes", *COAL Bulletim*, v. 21, pp. 2–7, 1992.

[48] LUCENA, A., BEASLEY, J., "A Branch-and-cut Algorithm for the Steiner Problem in Graphs", *Networks*, v. 31, pp. 39–59, 1998.

[49] MACULAN, N., "The STeiner Problem in Graphs", *Annals of Discrete Mathematics*, v. 31, pp. 185–212, 1987.

[50] MAGNANTI, T. L., WONG, T., "Network Design and Transportation Planning: Models and Algorithms", *Transportation Science*, v. 18, pp. 1–55, 1984.

[51] MANGASARIAN, O. L., SHIAU, T. H., "Lipschitz Continuity of Solutions of Linear Inequalities, Programs and Complementarity Problems", *SIAM Journal on Control and Optimization*, v. 25, pp. 583–595, 1987.

[52] MARTINS, S. L., RIBEIRO, C. C., SOUZA, M. C., "A Parallel GRASP for the Steiner Problem in Graphs", In: Proceedings of IRREGULAR'98 - 5th International Symposium on Solving Irregularly Structured Problems in Parallel, *Lecture Notes in Computer Science*, v. 1457, pp. 285–297, 1998.

[53] MARTINS, S. L., PARDALOS, P. M., RESENDE, M. G. C., RIBEIRO, C. C., "Greedy Randomized Adaptative Search Procedures for the Steiner Problem in Graphs", In: Rjasekaram,S., Pardalos, P. M., Rolim, J. (eds), *Randomization Methods in Algorithm Design*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science v. 43, pp. 133–145, Submetido para *Journal of Global Optimization*, 1998.

[54] MÜLLER, H., BRANDSTÅDT A., "The NP-completeness of STeiner Tree and Dominating Set for Chordal Bipartite Graphs", *Theoretical Computer Science*, v. 53, pp. 257–265, 1987.

[55] MURTY, K. G., *Linear Programming*. 1 ed. New York, John Wiley & Sons, 1983.

[56] PALMEIRA, M. M., LUCENA, A., PORTO, O., *A Relax and Cut Algorithm for the Quadratic Knapsack Problem* Submetido para publicação em *Discrete Applied Mathematics* em Dezembro de 1999.

[57] PORNAVALAI, C., SHIRATORI, N., CHAKRABORTY, G., "Neural Network for Optimal Steiner Tree Computation", *Neural Processing Letters*, v. 3, pp. 139–149, 1996.

[58] PRODON, A., LIEBLING, T. M., GROFFIN, H., *Steiner Problems on Two-trees*, Department de Mathématique, École Polytechnique Federale de Lausanne, Suisse, Technical Report RO 850315, 1985.

[59] RARDIN, R. L., PARKER, R. G., RICHEY, M. B., *A Polynomial Algorithm for a Class of Steiner Tree Problems on Graphs*, Department of Industrial and Systems Engineering, Georgia Institute of Technology, USA, Technical Report J-82-5, 1982.

[60] RAYWARD-SMITH, V. J., CLARE, A., "On Finding Steiner Vertices", *Networks*, v. 16, pp. 283–294, 1986.

[61] RIBEIRO, C. C., SOUZA, M. C., *Tabu Search for the Steiner Tree Problem in Graphs*, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Brasil, Submetido para publicação em *Networks*, 1997.

[62] ROBINSON, S. M., "Generalized Equations and Their Solutions, Part I: Basic Theory", *Mathematical Programming Study*, v. 10, pp. 128–141, 1979.

[63] ROBINSON, S. M., "Linear Convergence of ϵ-subgradient Descent Methods for a Class of Convex Functions", Aceito para publicação em *Mathematical Programming*, 1999.

[64] SHOR, N., *Minimization methods for non-differentiable functions.* 1 ed. Berlin, Springer-Verlag, 1985.

[65] SOLODOV, M. V., SVAITER, B. F., "A Hybrid Approximate Extragradient-proximal Point Algorithm Using the Enlargement of a Maximal Monotone Operator", Aceito para publicação em *Set-Valued Analysis*, 1999.

[66] SOUKUP, J., CHOW, W. F., "A Set of Test Problems for the Minimum Length Connection Networks", *ACM/SIGMAP Newsletters*, v. 15, pp. 48–51, 1973.

[67] TAKAHASHI, H., MATSUYAMA, A., "An Approxiamted Solution for the Steiner Tree Problem in Graphs", *Mathematica Japonica*, v. 254, pp. 573–577, 1980.

[68] VOß, S., "Steiner's Problem in Graphs: Heuristic Methods", *Discrete Applied Mathematics*, v. 40, pp. 45–72, 1992.

[69] WALD, J. A., COLBOURN, C. J., "Steiner Trees, Partial 2-trees and Mimimum IFI Networks", *Networks*, v. 13, pp. 159–167, 1983.

[70] WHITE, K., FARBER, M., PULLEYBLANK, W. R., "Steiner Trees, Connected Domination and Strongly Chordal Graphs", *Networks*, v. 15, pp. 109–124, 1987.

[71] WILLIAMSON, D. P., GOEMANS, M. X., MIHAIL, M., et al., "A Primal-dual Approximation Algorithm for Generalized Steiner Network Problems", *Combinatorica*, v. 15, n. 3, pp. :435–454, 1995.

[72] WINTER, P., "Reductions for the Rectilinear Steiner Tree Problem", *Networks*, v. 26, pp. 187–198, 1995.

[73] WINTER, P., SMITH, J. M., "Path-distance Heuristics for the Steiner Problem in Undirected Networks", *Algorithmica*, v. 7, pp. 309–327, 1992.

[74] WOLFE, P., "A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions", *Mathematical Programming Study*, v. 3, pp. 145–173, 1975.

[75] WOLSEY, L. A., *Integer Programming*. 1 ed. New York, John Wiley & Sons, 1998.

[76] WONG, R. T., "A Dual Ascent Approach for Steiner Tree Problems on a Directed Graph", *Mathematical Programming*, v. 28, pp. 271–287, 1984.

[77] XU, J., CHIU, S. Y., GLOVER, F., "A Probabilistic Tabu Search for the Telecommunications Network Design", *Combinatorial Optimization: Theory and Pratice*, v. 1, pp. 69–94, 1996.

[78] XU, J., CHIU, S. Y., GLOVER, F., "Using Tabu Search to Solve Steiner Tree-star Problems in Telecommunications Network Design", *Telecommunication Systems*, v. 6, pp. 117–125, 1996.