UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

Um Estudo Computacional de Cortes Derivados do Corte Chvátal-Gomory para Problemas de Programação Inteira

Autor: Sara Luísa de Andrade Fonseca
Orientador: Vinícius Amaral Armentano

Tese de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas - UNICAMP, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Banca Examinadora:
Prof. Dr. Vinícius Amaral Armentano - DENSIS/FEEC/ UNICAMP (orientador)
Prof. Dr. Carlos Eduardo Ferreira - DCC/IME /USP
Prof. Dr. Secundino Soares Filho - DENSIS/FEEC/UNICAMP
Prof. Dr. Takaaki Ohishi - DENSIS/FEEC/UNICAMP

Campinas, SP
2007

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Fonseca, Sara Luísa de Andrade
F733e Um estudo computacional de cortes derivados do corte Chvátal-Gomory para problemas de programação inteira / Sara Luísa de Andrade Fonseca, SP: [s.n.], 2007.

Orientador: Vinícius Amaral Armentano.
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Programação inteira. 2. Programação linear. 3. Pesquisa operacional. 4. Algoritmos. I. Armentano, Vinícius Amaral. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: A computational study of cuts derived from the Chvátal-Gomory cut for integer programming problems
Palavras-chave em Inglês: Integer programming, Valid inequalities, Cuts, Chvátal-Gomory, Branch-and-cut
Área de concentração: Automação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Carlos Eduardo Ferreira, Secundino Soares Filho, Takaaki Ohishi.
Data da defesa: 23/10/2007
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Sara Luísa de Andrade Fonseca

Data da Defesa: 23 de outubro de 2007

Título da Tese: "Um Estudo Computacional de Cortes Derivados do Corte Chvátal-Gomory para Problemas de Programação Inteira"

Prof. Dr. Vinícius Amaral Armentano (Presidente): Vinícius A. Armentano

Prof. Dr. Carlos Eduardo Ferreira: C. Ferreira

Prof. Dr. Takaaki Ohishi: T. Ohishi

Prof. Dr. Secundino Soares Filho: Secundino Soares F.

Resumo

Em 1958, Gomory propôs uma desigualdade válida ou corte a partir do tableau do método simplex para programação linear, que foi utilizado no primeiro método genérico para resolução de problemas de programação inteira. Em 1960, o corte foi estendido para problemas de programação inteira mista. Em 1973, Chvátal sugeriu um corte derivado da formulação original do problema de programação inteira, e devido à equivalência com o corte de Gomory, este passou a ser chamado de corte de Chvátal-Gomory. A importância do corte de Gomory só foi reconhecida em 1996 dentro do contexto do método *branch-and-cut* para resolução de problemas de programação inteira e programação inteira mista. Desde então, este corte é utilizado em resolvedores comerciais de otimização. Recentemente, diversos cortes novos derivados do corte de Chvátal-Gomory foram propostos na literatura para programação inteira. Este trabalho trata do desenvolvimento de algoritmos para alguns destes cortes, e implementação computacional em um contexto de *branch-and-cut*, no ambiente do resolvedor CPLEX. A eficácia dos cortes é testada em instâncias dos problemas da mochila multidimensional, designação generalizada e da biblioteca MIPLIB.

Palavras-chave: programação inteira, desigualdades válidas, cortes, Chvátal-Gomory, *branch-and-cut*.

Abstract

In 1958, Gomory proposed a valid inequality or cut from the tableau of the simplex method for linear programming, which was used in the first generic method for solving integer programming problems. In 1960, the cut was extended to handle mixed integer programming problems. In 1973, Chvátal suggested a cut that is generated from the original formulation of an integer programming problem, and due to the equivalence with the Gomory cut, it was named Chvátal-Gomory cut. The importance of the Gomory cut was recognized only in 1996 in the context of the branch-and-cut method for solving (mixed) integer programming problems. Today, such a cut is utilized in optimization commercial solvers. Recently, several new cuts derived from the Chvátal-Gomory cut have been proposed in the literature for integer programming. This work deals with the development of algorithms and computational implementations for some of the new proposed cuts, in a context of the branch-and-cut method, by using the CPLEX solver. The efficiency of the cuts is tested on instances of the multi-dimensional knapsack, generalized assignment problems, and instances from the MIPLIB library.

Keywords: integer programming, valid inequalities, cuts, Chvátal-Gomory, branch-and-cut.

Aos meus pais, Antônio e Magda,
que lutaram por mim em todos os sentidos;

Aos meus irmãos, Antônio e Lucianara,
por estarem presentes em minha vida.

Agradecimentos

Agradeço ao meu orientador Vinícius Amaral Armentano, pelos ensinamentos profissionais e pessoais, pelo conhecimento compartilhado e pela confiança demonstrada.

Agradeço aos meus pais Antônio e Magda, e aos meus irmãos, Antônio e Lucianara, por todo conforto emocional, incentivo e apoio, cujo valor é inestimável.

Agradeço ao Henrique, por tornar tudo mais leve.

Agradeço às minhas amigas, Andréa e Daiane, pela longas conversas. À minha amiga Maria Carolina, pelo apoio à longa distância.

Agradeço aos meus colegas de pós-graduação do DENSIS, pelas críticas e sugestões.

Agradeço aos médicos e do Hospital das Clínicas da UNICAMP por tornar tudo isso possível.

Agradeço ao CNPq, pelo apoio financeiro.

Sumário

Lista de Figuras **xiii**

Lista de Tabelas **xv**

Introdução **xvii**

1 Planos de Corte para Programação Inteira **1**
1.1 Conceitos Básicos 1
1.2 Cortes de Gomory 6
1.2.1 Corte de Gomory para Programação Inteira 8
1.2.2 Corte de Gomory para Programação Inteira Mista 11
1.3 Cortes Chvátal-Gomory 15
1.3.1 Equivalência entre o Corte de Chvátal-Gomory e o Corte Fracionário de Gomory 16
1.4 Algoritmo *branch-and-cut* 17
1.5 Medidas de Qualidade de Cortes 19

2 Descrição dos Cortes Implementados **23**
2.1 Cortes Genéricos em Programação Inteira 23
2.2 Corte Chvátal-Gomory Forte e Corte Fracionário Forte 24
2.2.1 Testes Computacionais 27
2.2.2 Exemplo 29
2.3 K-cortes 29
2.3.1 Testes Computacionais 33
2.4 Cortes Chvátal-Gomory-Nível 34
2.4.1 Variação de Parâmetros 40
2.4.2 Geração de Cortes Chvátal-Gomory-Nível 46
2.4.3 Exemplos 46
2.5 Exemplo Ilustrativo para os Três Cortes 50

3 Implementação Computacional e Resultados **53**
3.1 Método *branch-and-cut* 53
3.2 Geração de Cortes a partir do Tableau 54
3.3 Instâncias 57

3.3.1 Mochila Muldimensional . . . 57
3.3.2 Mochila Multidimensional com Restrições de Demanda . . . 57
3.3.3 Problema de Designação Generalizada . . . 58
3.3.4 MIPLIB 3.0 e MIPLIB 2003 . . . 59
3.4 CPLEX 10.1 . . . 59
3.4.1 Inserção de Cortes . . . 60
3.4.2 Plataforma de Testes . . . 60
3.5 Resultados Computacionais . . . 60
3.5.1 Uso de Reotimização . . . 60
3.5.2 Testes de Parâmetros . . . 64
3.5.3 Parâmetros Relativos a Cortes . . . 68
3.5.4 Corte Fracionário Forte . . . 68
3.5.5 K-cortes . . . 69
3.5.6 Corte Chvátal-Gomory-Nível . . . 69
3.6 Testes Comparativos . . . 70
3.7 Conclusões . . . 72

Referências bibliográficas **76**

A Complementação do Capítulo 2 **81**
A.1 Proposição 2.3 . . . 81
A.2 Proposição 2.4 . . . 85
A.3 Proposição 2.5 . . . 88
A.4 Proposição 2.7 . . . 90

Lista de Figuras

1.1 Três diferentes formulações para o conjunto X. 3
1.2 Região factível para o exemplo 1.1 4
1.3 Adição de desigualdades válidas ao exemplo 1.1. 5
1.4 Relação de dominância: coeficientes positivos. 7
1.5 Relação de dominância entre desigualdades. 8
1.6 Árvore gerada pela aplicação do método *branch-and-cut* ao exemplo 1.4 20
1.7 Aplicação de *branch-and-cut* ao exemplo 1.4 20

2.1 Comparação entre desigualdades CGF e CG. 30
2.2 Comparação entre k-cortes e GIF no exemplo 2.2 32
2.3 γ_j/γ_0 em função de f_j para GI, GIF e 5-corte 33
2.4 a^p em função de d para $a > 0$ 38
2.5 a^p em função de d para $a \leq 0$ 39
2.6 Desigualdades CG-Nível e corte GIF 49
2.7 Aplicação do algoritmo de 2 fases. 50
2.8 Comparação entre cortes CFF com e sem pré-multiplicação por -1. 52
2.9 Comparação entre cortes CFF, CG-Nível e K-cortes. 52

3.1 Número de nós para a instância b05100 63
3.2 Número de cortes inseridos para a instância b05100 63
3.3 % Gap fechado para a instância b05100 64

Lista de Tabelas

3.1 Parâmetros usados no teste de comparação entre IC e IR 61
3.2 Comparação entre as técnicas IC e IR 62
3.3 % Gap fechado com variação tamanho do *round* 65
3.4 Variação do tamanho do *round* 66
3.5 % Gap fechado para os critérios de seleção violação normalizada e eficiência 66
3.6 % Gap fechado com e sem uso de descarte por paralelismo 67
3.7 Uso de descarte por paralelismo 67
3.8 Variação do máximo k para o corte CFF 68
3.9 Variações do máximo de cortes por linha para o k-corte 69
3.10 Parâmetros de teste para ALG-CGN 69
3.11 Resultados de teste para ALG-CGN 70
3.12 Comparação entre o uso de 1 ou 2 fases ($p_{max} = 3$, MAXPY $= 3$) 71
3.13 Parâmetros utilizados nos testes da Seção 3.6 71
3.14 % Gap fechado 74
3.15 Média do gap fechado por tipo de instância 75
3.16 Médias para cada plano de corte 75
3.17 Razão de cortes por nó 75

Introdução

No artigo “Early Integer Programming” (Gomory, 2002), Ralph E. Gomory relata suas experiências pioneiras na área de programação inteira, ainda não nascida. Gomory obteve o título de PhD em Matemática na universidade de Princeton em 1954, com trabalho em equações diferenciais não lineares, sob a orientação do renomado Prof. Solomon Lefschtez.

Em 1955, passou a trabalhar no Physics Branch of the Office of Naval Research em Washington e, neste ano, conheceu pessoas do Operations Research Group, iniciando seu conhecimento em pesquisa operacional. No outono de 1957, tornou-se professor do departamento de Matemática da universidade de Princeton, onde era dirigido por A. W. Tucker e contava com Harold Kuhn e Martin Beale, pesquisadores que tiveram uma contribuição expressiva na área de otimização.

Gomory permaneceu como consultor da marinha em Washington, e, em uma dessas visitas, assistiu a uma apresentação de um modelo de programação linear. Um dos apresentadores destacou que seria interessante ter um número inteiro em lugar de 1,3 aviões, que não tinha significado.

Gomory colocou então, como objetivo, inventar um método que produzisse resultados inteiros. Passou uma semana tentando combinar métodos de solução de equações lineares com variáveis inteiras (equações diofantinas) e programação linear, sem sucesso. No oitavo dia, raciocinou da seguinte forma em relação a um problema de programação inteira com coeficientes inteiros na função objetivo: se tivesse que obter uma solução inteira, então o primeiro passo seria obter o valor ótimo da função objetivo a ser maximizada através de programação linear, por exemplo, $7\frac{1}{4}$, e daí concluir que o valor máximo inteiro é 7.

A partir dessas idéias, Gomory chegou a duas conclusões importantes. Em primeiro lugar o valor ótimo dado por programação linear é um limitante superior para o valor ótimo de uma solução inteira. Em segundo lugar, se os coeficientes da função objetivo são inteiros, então é válido adicionar a desigualdade linear impondo que a função objetivo ≤ 7.

Esta desigualdade em que um número real é $\geq$ ao maior inteiro contido neste número (ou $\leq$ ao menor inteiro que contém este número) foi a base para Gomory inventar o método dos cortes fracionários (Gomory, 1958) e sua demonstração de finitude (Gomory, 1963a). Este corte é construído a partir do tableau ótimo da solução do problema linear, correspondente à relaxação de integralidade.

Depois de mais de um mês de trabalho, Gomory produziu um código do algoritmo em Fortran e passou a testar problemas de grande porte para a época, envolvendo dez a quinze variáveis. A maioria das instâncias eram resolvidas rapidamente, mas em uma delas um grande número de cortes foi adicionado sem chegar à solução inteira ótima. No verão de 1959, Gomory passa a trabalhar na IBM Research e, com maior suporte computacional, passa a experimentar a imprevisibilidade dos resultados computacionais que constantemente ocorriam. Em 1960, o corte fracionário é estendido para programação inteira mista, que gerou um corte fracionário mais forte para programação inteira.

Em 1963, Gomory apresentou um algoritmo para programação inteira que envolve somente números inteiros (Gomory, 1963b). Desigualdades válidas foram adicionadas ao método dual simplex de modo que sempre existe um elemento pivot igual a 1 pelas regras normais de escolha do pivot. Infelizmente, os tempos computacionais eram superiores aos do método fracionário, além de gerar valores inteiros muito grandes no tableau.

Estes experimentos computacionais não bem sucedidos foram o primeiro sinal de que o corte fracionário de Gomory "não funcionava". Nas décadas de 70 e 80 não foram publicados artigos reportando experimentos computacionais abrangentes com os cortes fracionários. Padberg e Rinaldi (1991) fizeram as seguintes afirmações sobre os cortes de Gomory: "Estes planos de corte têm propriedades de convergência fraca...planos de corte clássicos geram cortes fracos". "Um casamento de planos de corte e busca em árvore está fora de questão como uma forma de solução de problemas de otimização combinatória de grande porte".

Estas afirmações refletiam um pensamento unânime da comunidade científica (Cornuéjols, 2007) no início da década de 90, em que para resolver problemas de programação inteira de tamanho significativo tinha-se que explorar a estrutura do problema combinatório. Os cortes de Gomory geraram uma teoria elegante, mas não tinham utilidade na prática, porque eram cortes genéricos e não exploravam a estrutura. Algoritmos de *branch-and-cut* para problemas estruturados tiveram grande impacto. Padberg e Rinaldi (1991) obtiveram excelentes resultados para o problema do caixeiro viajante. O sucesso do *branch-and-cut* em um grande número de classes de problemas neste período era atribuído à abordagem de resolução estruturada, que envolvia a identificação de facetas da envoltória convexa (*convex hull*), algoritmos exatos ou heurísticos de separação e resultados computacionais.

No início dos anos 90, Balas, Céria e Cornuéjols (1993, 1996a) propuseram outra categoria de cortes genéricos denominados *lift-and-project* para problemas mistos com variáveis binárias, e demonstraram que tais cortes dominam os cortes de Gomory. Estes cortes mostraram-se fortes em instâncias difíceis, contrariando a visão convencional que cortes fortes teriam que ser projetados para explorar a estrutura do problema. Estes resultados provocaram grande curiosidade em Cornuéjols (2007), que solicitou à Céria, então seu orientado de doutorado, que implementasse os cortes de Gomory, que, embora mais fracos, eram mais rapidamente computados.

Os resultados computacionais obtidos com a implementação de Céria foram surpreendentes. A apresentação de Céria em uma conferência em 1994 deixou vários pesquisadores renomados da área perplexos. Os bons resultados obtidos com os cortes *lift-and-project* também foram obtidos com os cortes de Gomory. Além disso, o método *branch-and-cut* baseado em cortes de Gomory era mais rápido e robusto que o método *branch-and-bound*.

Em 1996, Balas, Ceria, Cornuéjols e Natraj (Balas et al., 1996b) demonstraram que os cortes de Gomory para problemas mistos com variáveis binárias são válidos em todos os nós da árvore de enumeração, permitindo que cortes gerados em nós distintos possam ser compartilhados. Os resultados obtidos com instâncias da MIPLIB foram novamente surpreendentes em termos de número de instâncias resolvidas e rapidez em relação ao *branch-and-bound*.

Os autores destacam três razões para o bom funcionamento dos cortes de Gomory: resolvedores de programação linear mais robustos, adição de todos os cortes do tableau (em lugar de um corte, como tentado por Gomory) e uso na estrutura *branch-and-cut* (em lugar de um enfoque de cortes puro). Este tipo de resultado, certamente teve grande influência para a inclusão de cortes de Gomory nos pacotes comerciais de otimização CPLEX, XPRESS e LINDO, e também para renovar o interesse de pesquisadores por cortes baseados em cortes de Gomory.

Em um artigo importante, Chvátal (1973) propôs um procedimento que, aplicado por um número finito de vezes (denominado *rank*), gera todas as desigualdades válidas da envoltória convexa das soluções de um problema de programação inteira. Estas desigualdades são obtidas por combinação linear ponderada das linhas da matriz da formulação original e arredondamento de coeficientes das linhas. Chvátal também mostrou implicitamente que qualquer corte fracionário de Gomory é equivalente à aplicação de uma vez do procedimento de Chvátal (*rank* um). Devido a esta equivalência o corte passou a ser chamado de Chvátal-Gomory.

O texto acima é o pano de fundo para esta dissertação cujo objetivo é implementar computacionalmente três tipos de cortes propostos recentemente para programação inteira que têm como base os cortes de Chvátal-Gomory.

O primeiro corte foi proposto por Lechtford e Lodi (2002) e envolve o fortalecimento do corte Chvátal-Gomory, podendo originar cortes de rank dois. Neste artigo, testes computacionais limitados são apresentados para este corte e outro derivado do corte fracionário de Gomory.

O segundo corte (Cornuéjols, Li e Vandenbussche, 2003) envolve a multiplicação de uma linha do tableau por um número inteiro seguido pelo procedimento de Gomory para obtenção do corte fracionário inteiro forte derivado do corte fracionário para programação inteira mista. Neste trabalho são apresentados resultados teóricos de dominância em relação ao corte fracionário ordinário e ao corte forte de Gomory, e testes computacionais limitados.

Finalmente, Glover e Sherali (2005) propuseram uma nova classe de cortes denominada Chvátal-Gomory com níveis (*Chvátal-Gomory-tier*). Estes cortes são obtidos pela aplicação do procedimento Chvátal-Gomory a partir de uma linha da matriz da formulação original que é escalada por um parâmetro d. O mesmo processo de geração do corte é então repetido p vezes (incluindo a primeira aplicação), enquanto o parâmetro de escalamento é reduzido de uma unidade em cada aplicação, de forma que $1 \leq p \leq d$. O nível de camada p pode ser especificado através de restrições impostas aos coeficientes do corte, e, para cada camada p, é possível determinar o corte mais forte de acordo com um certo critério. Este enfoque gera uma variedade de cortes fortes, incluindo cortes que não podem ser gerados por cortes de Chvátal-Gomory de *rank* um.

A dissertação está dividida em três capítulos. No primeiro capítulo são apresentadas as derivações dos cortes fracionários de Gomory para programação inteira e programação inteira mista, e dos cortes de Chvátal-Gomory. A conexão entre estes cortes também é exposta. O capítulo 2 descreve a derivação dos três tipos de cortes acima mencionados, envolvendo demonstração de validade e exemplos ilustrativos da força destes cortes. Finalmente, o capítulo 3 contém os experimentos computacionais com os três tipos de corte. Estes experimentos envolvem a programação dos cortes e da conexão com o método *branch-and-bound* (sem cortes e heurísticas) do pacote CPLEX 10.0, além de estratégias de escolha de cortes.

Capítulo 1

Planos de Corte para Programação Inteira

Neste capítulo são apresentados definições e conceitos básicos de programação inteira mista (PIM), programação inteira (PI) e cortes. São apresentados os planos de corte de Gomory e Chvátal-Gomory, bem como a relação entre eles. Ao final, apresenta-se o método *branch-and-cut* e medidas de qualidade de cortes.

1.1 Conceitos Básicos

Um *programa inteiro misto* com n variáveis inteiras, p variáveis reais e m restrições é definido como

$$\begin{array}{lll} (PIM) & \max & cx + dy \\ & \text{sujeito a} & Ax + Gy \leq b \\ & & x \in Z_+^n, y \in R_+^p, \end{array}$$

em que R_+^p é o espaço dos vetores de dimensão p com componentes reais não negativas, Z_+^n é o espaço dos vetores de dimensão n com componentes inteiras não negativas, A é uma matriz $(m \times n)$, G é uma matriz $(m \times p)$, b é um vetor $(m \times 1)$, c é um vetor $(1 \times n)$ e d é um vetor $(1 \times p)$.

Se todas as variáveis são reais, tem-se *o programa linear* dado por

$$\begin{array}{lll} (PL) & \max & dy \\ & \text{sujeito a} & Gy \leq b \\ & & y \in R_+^p. \end{array}$$

Se todas as variáveis são inteiras, define-se o *programa inteiro*

$$\begin{array}{lll} (PI) & \max & cx \\ & \text{sujeito a} & Ax \leq b \\ & & x \in Z_+^n. \end{array}$$

O problema inteiro com variáveis binárias é um caso especial dos problemas acima, definido sobre B^n, o espaço dos vetores de dimensão n com componentes binários ou 0-1.

O conjunto de restrições lineares de PL define em R^p um conjunto convexo, como colocado a seguir.

Definição 1.1. O subconjunto de R^p descrito por um conjunto de desigualdades lineares $P = \{Gy \leq b, y \in R^p\}$ é um *poliedro*. □

A seguir é formalizada a definição de uma formulação para um problema com variáveis inteiras que se baseia no fato de que a região factível deve conter todos os pontos inteiros factíveis.

Definição 1.2. Um poliedro $P \subseteq R^{n+p}$ é uma *formulação* para um conjunto $X \subseteq Z^n \times R^p$, se e somente se $X = P \cap (Z^n \times R^p)$. □

Métodos de solução para programação inteira, como *branch-and-bound*, resolvem problemas do tipo PI ou PIM através da resolução de sub-problemas em que são desconsideradas as restrições de integralidade sobre as variáveis. Portanto, estes métodos atuam na resolução de *relaxações lineares* do problema original.

Definição 1.3. Seja o problema inteiro misto com formulação P, dado por $\max\{cx + dy : x \in P \cap Z^n_+, y \in P \cap R^p_+\}$. A *relaxação linear* correspondente é dada pelo programa linear, $\max\{cx+dy : (x, y) \in P \cap (R^{n+p}_+)\}$. □

Formulações distintas de PI ou PIM podem definir diferentes relaxações lineares. Considere o seguinte conjunto de pontos $X = \{(1,1),(2,1),(3,1),(1,2),(2,2),(3,2),(2,3)\}$. A Figura 1.1 mostra três possíveis formulações para este conjunto. Apesar de ambas conterem todos os pontos inteiros válidos, a formulação P_3 define um espaço de soluções factíveis menor que P_1 e P_2.

A formulação P_3 tem como pontos extremos, apenas pontos inteiros. Ela constitui a *envoltória convexa* do conjunto X, como definido a seguir.

Definição 1.4. Dado um conjunto $X \subseteq R^n$, a *envoltória convexa* de X é o conjunto conv$(X) = \{x : x = \sum_{i=1}^k \lambda_i x^i, \sum_{i=1}^k \lambda_i = 1, \lambda \in R^k_+$, para todos os subconjuntos finitos $\{x^1, ..., x^k\} \in X\}$. □

A proposição a seguir é demonstrada em (Nemhauser e Wolsey, 1988).

Proposição 1.1. Seja o conjunto $X = \{(x, y) \in Z^n_+ \times R^p_+ : Ax + Gy \leq b\}$, em que A, G, b são racionais. A *envoltória convexa* do conjunto X, representada por conv(X) = $\{(x, y) : \tilde{A}x + \tilde{G}y \leq \tilde{b}, (x, y) \in R^{n+p}_+\}$, é um poliedro e os pontos extremos de conv(X) estão contidos em X. □

Portanto, em teoria, o problema PIM pode ser reformulado como um problema linear

$$(\tilde{P}) = \quad \max\{cx : \tilde{A}x + \tilde{G}y \leq \tilde{b}, (x, y) \in R^{n+p}_+\}.$$

Entretanto, em geral, a determinação da envoltória convexa de X é um problema de difícil solução, e uma aproximação da mesma pode ser buscada através da derivação de desigualdades válidas para X. Desigualdades válidas são aquelas que, quando adicionadas à formulação original, não eliminam nenhum ponto da região factível de PIM e têm a função de diminuir a região factível dada pela relaxação linear de PIM.

Definição 1.5. Uma desigualdade $\pi x \leq \pi_0$ é uma desigualdade válida para $X \subseteq R^n$ se $\pi x \leq \pi_0$ para todo $x \in X$. □

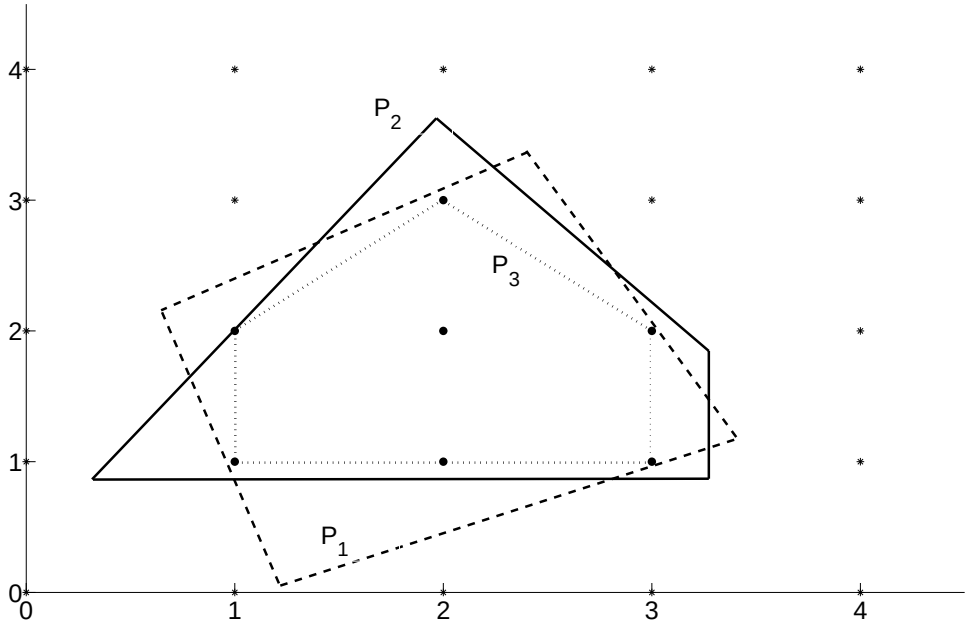

Figura 1.1: Três diferentes formulações para o conjunto X.

Desigualdades válidas utilizam, freqüentemente, o arredondamento de valores reais. Portanto são definidos a seguir os operadores de piso e teto de um número real.

Definição 1.6. Para qualquer número $r \in R$, define-se o *maior inteiro menor ou igual a* r como $\lfloor r \rfloor$(piso de r). Analogamente, define-se o *menor inteiro maior ou igual a* r como $\lceil r \rceil$(teto de r). A *parte fracionária* de r é definida como a função $f(r) = r - \lfloor r \rfloor$. □

A definição acima é estendida para um vetor ao se considerar o piso e o teto para todos os componentes deste vetor.

Duas desigualdades válidas clássicas são mostradas a seguir. Elas se baseiam unicamente no fato de que uma variável inteira não pode assumir valores reais.

Proposição 1.2. Seja $X^1 = \{y \in Z : y \leq b\}$ e $X^2 = \{y \in Z : y \geq b\}$. Então a desigualdade $y \leq \lfloor b \rfloor$ é válida para X^1, e a desigualdade $y \geq \lceil b \rceil$ é válida para X^2. □

As desigualdades acima são a base para a geração de cortes de Gomory, descritos na próxima seção. Planos de corte são desigualdades válidas que têm a função de diminuir a região factível da relaxação linear de um PIM. Dado que $X = \{(x, y) \in Z_+^n \times R_+^p : Ax + Gy \leq b\}$, um corte relativo a um ponto não pertencente a conv(X) mas pertencente à relaxação linear de X é uma desigualdade válida que elimina este ponto.

O exemplo a seguir ilustra o uso de desigualdades válidas que possibilitam a construção da envoltória convexa a partir da formulação inicial. Neste exemplo fica claro que a introdução de desigualdades válidas acarreta diminuição da região factível e do valor da função objetivo. Como regra

geral, se denominarmos X_{PI}, X_{PIM} e X_{PL}, os conjuntos de soluções factíveis de PI, PIM e PL, então, como $Z^n_+ \subset R^n_+$, temos $X_{PI} \subset X_{PIM}$ e $X_{PIM} \subset X_{PL}$. Portanto, o valor ótimo da função objetivo z^* será tal que $z^*_{PI} \leq z^*_{PIM} \leq z^*_{PL}$.

Exemplo 1.1. Considere o problema de maximização a seguir (Garfinkel e Nemhauser, 1972).

$$\begin{array}{ll} \max & 2x_1 + x_2 \\ \text{s.a.} & x_1 + x_2 \leq 5 \\ & -x_1 + x_2 \leq 0 \\ & 6x_1 + 2x_2 \leq 21 \\ & x \in Z^2_+. \end{array}$$

A região que representa a relaxação linear deste problema está na Figura 1.2. Os pontos em destaque são as soluções possíveis para este PI. As retas pontilhadas representam os pontos em que a função objetivo assume valores constantes.

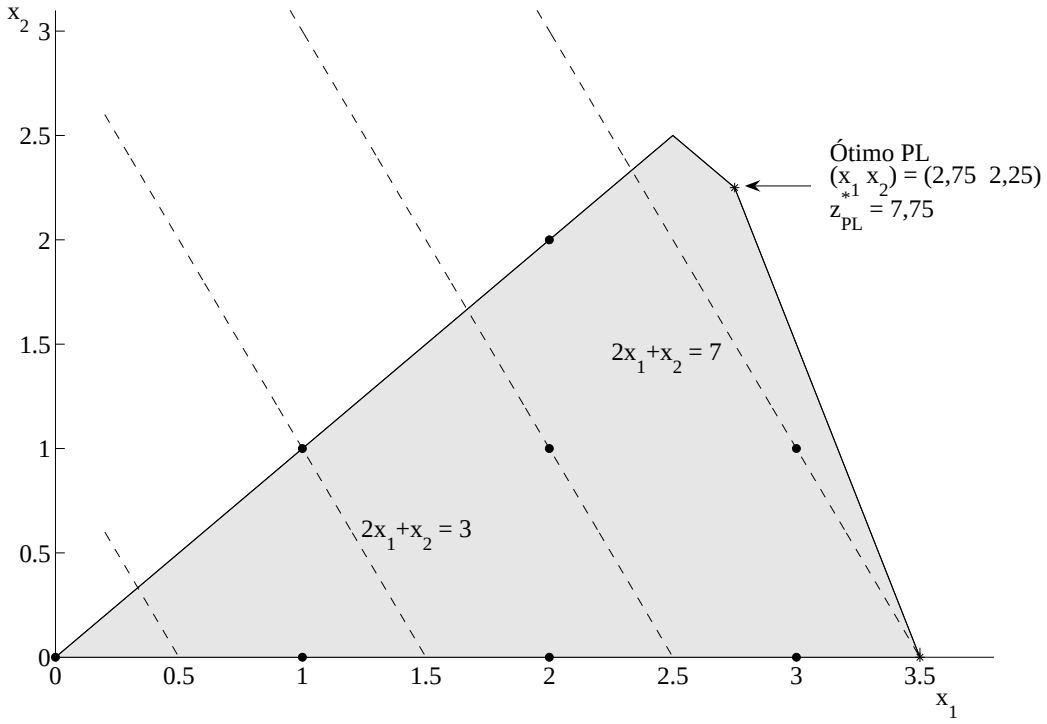

Figura 1.2: Região factível para o exemplo 1.1

As restrições de integralidade sobre x_1 e x_2 podem ser utilizadas na geração de desigualdades válidas. Por exemplo, a Proposição 1.2 pode ser aplicada. Como $x_1 \leq 3,5$ e $x_1 \in Z_+$, a desigualdade $x_1 \leq 3$ é imediatamente derivada. Se adicionarmos ao problema inicial uma segunda desigualdade dada por $x_1 + x_2 \leq 4$, construímos a envoltória convexa deste problema, representada na Figura 1.3.

Esta desigualdade é derivada por observação da região factível. Para problemas de maior dimensão, é necessário adotar um método sistemático de derivação de desigualdades.

Observe que a solução ótima do problema linear da Figura 1.3 é um ponto inteiro, pois as restrições lineares representam a envoltória convexa dos pontos inteiros factíveis. Assim, temos, na Figura 1.2, o ponto ótimo fracionário $(x_1, x_2) = (2,75; 2,25)$, com valor de função objetivo $z^*_{PL} = 7,75$, enquanto que a Figura 1.3 mostra o ponto ótimo inteiro $(x_1, x_2) = (3, 1)$, com valor de função objetivo $z^*_{PI} = 7$. □

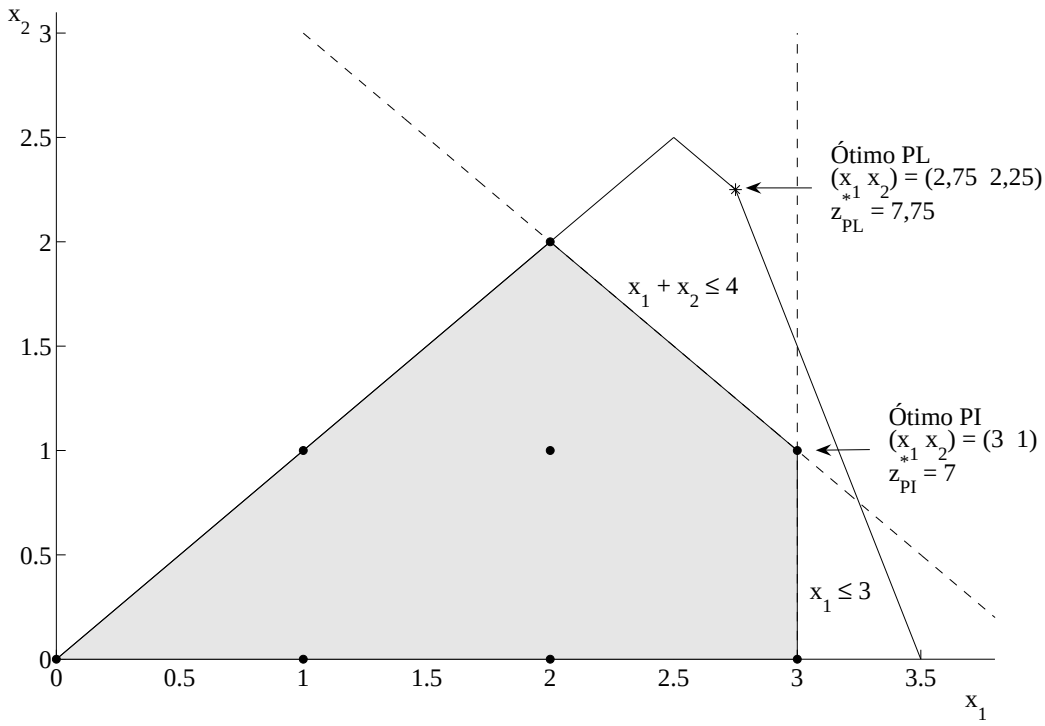

Figura 1.3: Adição de desigualdades válidas ao exemplo 1.1.

Faces e Facetas

Algumas desigualdades podem constituir faces ou facetas do poliedro P. Considere que o poliedro $P \in R^n$ possui n direções linearmente independentes, ou seja, tem *dimensão completa*. As definições a seguir são baseadas em (Wolsey, 1998).

Definição 1.7. O conjunto de pontos $x^1, ..., x^k \in R^n$ é *afim independente* se as $k-1$ direções dadas por $x^2 - x^1, ..., x^k - x^1$ são linearmente independentes. □

Definição 1.8. A *dimensão* do poliedro P, $dim(P)$ é o máximo de pontos afim independentes em P menos um. □

Portanto, o poliedro $P \in R^n$ tem dimensão completa se e somente se $dim(P) = n$.

Definição 1.9. F define uma *face* do poliedro P se $F = \{x \in P : \pi x = \pi_0\}$ para uma desigualdade válida de P, $\pi x \leq \pi_0$. F é uma *faceta* do poliedro P se F é uma face de P de dimensão $dim(F) = dim(P) - 1$. □

No exemplo 1.1, o poliedro definido no R^2 é de dimensão completa, pois os pontos $(2, 0)$, $(2, 1)$ e $(3, 1)$ são afim independentes, e portanto $dim(P) = 2$. As desigualdades $x_1 \leq 3$ e $x_1 + x_2 \leq 4$ definem facetas do problema inteiro, pois constituem faces de P de dimensão $n - 1 = 1$. Facetas são necessárias para a descrição de um poliedro (Nemhauser e Wolsey, 1988).

Comparação de Desigualdades

Outra desigualdade válida para o exemplo 1.1 é $x_1 \leq 3{,}5$. Neste caso, fica claro que esta desigualdade é menos restritiva que a desigualdade $x_1 \leq 3$. Entretanto, em geral esta comparação não é direta. É necessário definir uma regra de comparação de desigualdades, de forma a determinar se uma delas é mais restritiva, ou mais forte. Para tanto, define-se sob que condições diz-se que uma desigualdade, ou um corte, *domina* outro.

Definição 1.10. As desigualdades válidas $\pi x \leq \pi_0$ e $\gamma x \leq \gamma_0$ são equivalentes se $(\gamma, \gamma_0) = \lambda(\pi, \pi_0)$, para um dado $\lambda > 0$. Se elas não forem equivalentes e existir $\mu > 0$ tal que $\gamma \geq \mu\pi$ e $\gamma_0 \leq \mu\pi_0$, então $\{x \in R^n_+ : \gamma x \leq \gamma_0\} \subset \{x \in R^n_+ : \pi x \leq \pi_0\}$. Neste caso, $\gamma x \leq \gamma_0$ domina ou é mais forte que $\pi x \leq \pi_0$. De forma equivalente, pode-se dizer que a desigualdade $\pi x \leq \pi_0$ é dominada por, ou é mais fraca que $\gamma x \leq \gamma_0$. □

Note que a relação $\subset$ implica que um conjunto está estritamente contido no outro. Note ainda que se existe $\mu > 0$ tal que $\gamma \geq \mu\pi$ e $\gamma_0 \leq \mu\pi_0$, então $\gamma x \leq \gamma_0$ implica que $\mu\pi x \leq \gamma_0 \leq \mu\pi_0$, que implica $\pi x \leq \pi_0$.

Exemplo 1.2. Considere duas desigualdades $4x_1 + 4x_2 \leq 15$ e $10x_1 + 5x_2 \leq 16$. Seja $(\gamma, \gamma_0) = (10, 5, 16)$ e $(\pi, \pi_0) = (4, 4, 15)$. Note que $(10, 5) \geq \mu(4, 4)$, para qualquer $\mu \leq 5/4$, e $16 \leq 15\mu$, para qualquer $\mu \geq 16/15$. Então, para $\mu = 16/15$, $\gamma \geq \mu\pi$ e $\gamma_0 \leq \mu\pi_0$. A Figura 1.4 mostra que a primeira inequação é dominada pela segunda.

Considere, agora, as desigualdades $-2x_1 + x_2 \leq 1$ e $-x_1 + x_2 \leq 1$. Seja $(\gamma, \gamma_0) = (-1, 1, 1)$ e $(\pi, \pi_0) = (-2, 1, 1)$. Para $\mu = 1$, $(-1, 1) \geq \mu(-2, 1)$ e $1 \leq 1\mu$. A Figura 1.5 mostra que a primeira inequação é dominada pela segunda. □

1.2 Cortes de Gomory

Em 1958, Gomory introduz os *cortes fracionários* para resolução de problemas de programação inteira (Gomory, 1958) e (Gomory, 1960b). Estes dois trabalhos introduzem seu método de solução de problemas com variáveis inteiras, formalizado posteriormente em 1963 (Gomory, 1963a). Neste trabalho, Gomory mostra que é possível resolver um problema de programação inteira apenas adicionando cortes fracionários, reotimizando o programa linear e repetindo este processo por um número finito de vezes, até que a solução inteira ótima seja encontrada. O método é denominado método de formas inteiras (*The Method of Integer Forms*). Para demonstrar que a inserção de suas desigualdades

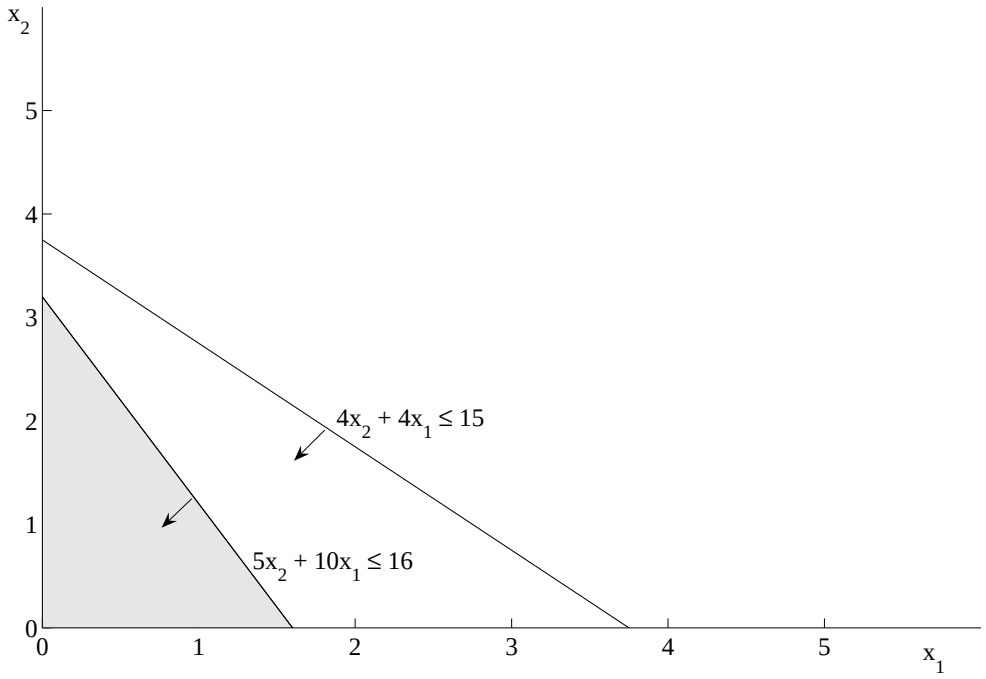

Figura 1.4: Relação de dominância: coeficientes positivos.

pode ser iterativa, o autor mostra que as variáveis de folga provenientes da inserção de seus cortes também são inteiras. Este corte é detalhadado na seção 1.2.1.

Em 1960, Gomory estende seu trabalho para programação inteira mista (Gomory, 1960a). O corte de Gomory para PIM pode ser particularizado para PI. Este novo corte fracionário é mais forte que o original. A seção 1.2.2 apresenta a dedução destes cortes.

Em 1963, Gomory propõe um método para solução de problemas com variáveis inteiras que utiliza apenas coeficientes inteiros durante todo o processo de otimização (Gomory, 1963b). Este método ficou conhecido como *All-Integer Algorithm* (algoritmo apenas com inteiros). Neste método, o elemento escolhido para pivoteamento (pelas regras padrões) é sempre igual a 1. Os cortes são adicionados ao método dual simplex de modo a viabilizar este pivoteamento. Valores muito grandes no tableau resultante são um efeito colateral não desejado. Resultados computacionais mostram que este método é inferior ao que utiliza os cortes fracionários.

Os cortes de Gomory foram considerados ineficientes na prática até os meados dos anos 90, quando foram associados com sucesso ao método *branch-and-bound* por Balas et al. (1996b). Até então, apenas duas publicações reportavam sucesso na aplicação dos cortes de Gomory, mas envolviam uso de informações a respeito da estrutura do problema 0-1. Martin (1963) utiliza cortes fracionários em seu Algoritmo Euclideano para programação inteira. Miliotis (1978) resolve o problema do caixeiro viajante aplicando cortes fracionários de Gomory em um processo iterativo em que são adicionadas também restrições de sub-rota. Cortes são adicionados ao problema linear de designação repetidamente até que uma solução inteira seja obtida. Se esta solução for uma solução factível para o problema do caixeiro viajante, o processo termina com a solução ótima inteira. Caso contrário, são

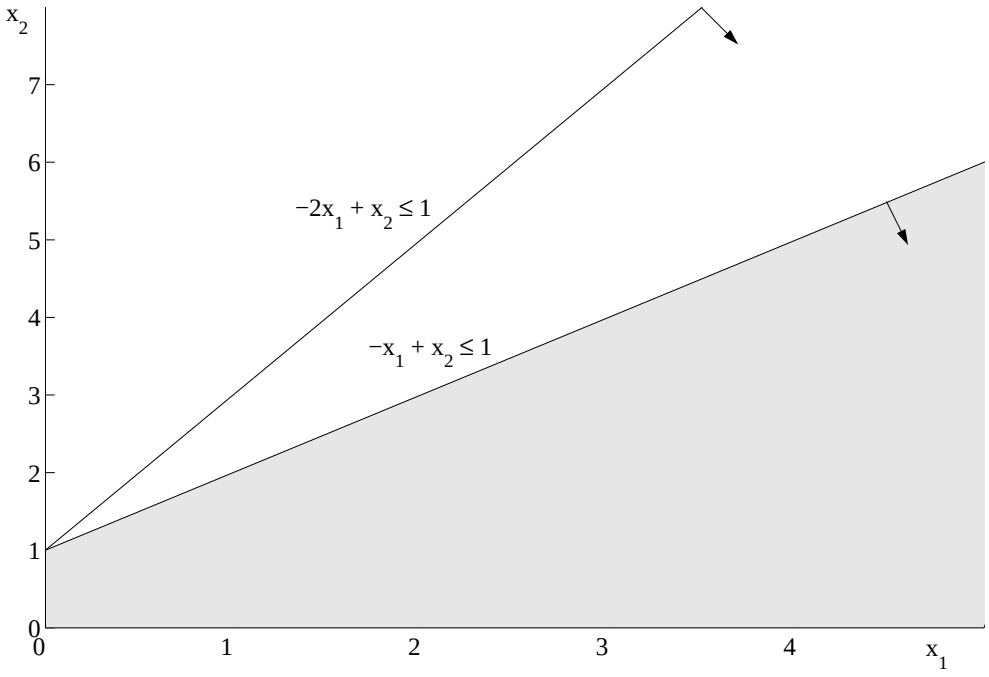

Figura 1.5: Relação de dominância entre desigualdades.

geradas restrições de sub-rota que eliminem a solução atual e o processo é repetido. Um algoritmo análogo é proposto tal que a ordem de uso de restrições de sub-rota e cortes de Gomory é trocada.

O sucesso obtido por Balas et al. (1996b) é atribuído ao fato de que os resolvedores da década de noventa já eram numericamente mais estáveis do que aqueles utilizados nos primeiros testes dos cortes de Gomory. Esta não é, porém, a única razão apontada como causa do sucesso da aplicação dos cortes.

Após a reotimização, são removidos do PL cortes que apresentam variáveis de folga básicas. A variável de folga básica indica que este corte não está ativo na solução atual. Desta forma, o problema linear mantém-se em um tamanho razoável e, além de propiciar uma maior estabilidade numérica, pode ser resolvido mais rapidamente.

Outro fator importante é a validade global dos cortes de Gomory. Os autores demonstram como os cortes gerados em um determinado nó da árvore de *branch-and-bound* podem ser utilizados em qualquer outro nó da árvore e destacam que a utilização dos cortes apenas localmente gera piores resultados. Na seção 1.2.2, a validade global do corte de Gomory para programação inteira mista é demonstrada.

1.2.1 Corte de Gomory para Programação Inteira

Os cortes de Gomory são construídos a partir de linhas do tableau resultante da resolução da relaxação linear do programa inteiro.

Seja o problema de programação inteira após a inclusão de variáveis de folga, se necessário:

$$\begin{array}{lll} (PI) & \max & cx \\ & \text{sujeito a} & Ax = b \\ & & x \in Z_+^n. \end{array}$$

Seja J o conjunto das variáveis não-básicas e I o conjunto das variáveis básicas associadas à solução ótima do problema de programação linear relativo à relaxação linear de PI. A i-ésima linha do tableau ótimo é representada por

$$x_i = \bar{a}_{i0} + \sum_{j \in J} \bar{a}_{ij}(-x_j), \forall\, i \in I. \tag{1.1}$$

Os coeficientes de (1.1) podem ser reescritos como

$$\bar{a}_{ij} = \lfloor \bar{a}_{ij} \rfloor + f_{ij}, \forall\, i \in I \text{ e } \forall\, j \in J,$$
$$\bar{a}_{i0} = \lfloor \bar{a}_{i0} \rfloor + f_{i0}, \forall\, i \in I,$$

tal que $0 \leq f_{ij} < 1$ e $0 \leq f_{i0} < 1$.

Para facilitar a notação, o índice i na equação (1.1), relativa à i-ésima linha do tableau, é suprimido. Assim temos $\bar{a}_{ij} = \bar{a}_j$ e $f_{ij} = f_j$.

Admita que a solução ótima não é inteira. Então existe uma linha i do tableau associado à solução ótima $x_i + \sum_{j \in J} \bar{a}_j x_j = \bar{a}_0$ tal que $\bar{a}_0$ é não inteiro.

A desigualdade $x_i + \sum_{j \in J} \lfloor \bar{a}_j \rfloor x_j \leq \bar{a}_0$ é válida em PI, pois $x_j \geq 0$. Como o lado esquerdo da desigualdade anterior é inteiro, segue-se que

$$x_i + \sum_{j \in J} \lfloor \bar{a}_j \rfloor x_j \leq \lfloor \bar{a}_0 \rfloor \tag{1.2}$$

também é válida em PI. Eliminando x_i através de (1.1), tem-se:

$$\sum_{j \in J} (\bar{a}_j - \lfloor \bar{a}_j \rfloor) x_j \geq \bar{a}_0 - \lfloor \bar{a}_0 \rfloor$$

ou

$$\sum_{j \in J} f_j x_j \geq f_0\,, \tag{1.3}$$

que corresponde ao *corte fracionário* de Gomory (GI), em que $0 \leq f_j < 1$ e $0 < f_0 < 1$. Como $x_j^* = 0$ para todo $j \in J$, a desigualdade acima corta a solução básica $x_i = \bar{a}_0$.

Adicionando-se a variável de folga, o corte de Gomory pode ser reescrito como

$$s = -f_0 + \sum_{j \in J} f_j x_j, \quad s \geq 0.$$

Como

$$x_i = -(-f_0 + \sum_{j \in J} f_j x_j) + (\lfloor \bar{a}_0 \rfloor - \sum_{j \in J} \lfloor \bar{a}_j \rfloor x_j),$$

segue-se que a variável de folga s é inteira. Este fato fica claro também através da desigualdade (1.2). Como todos os coeficientes são inteiros, assim como todas as variáveis x_j e x_i, então a variável de folga deve ser inteira.

Duas diferentes formas de representar o corte fracionário de Gomory são expostas nas equações (1.2) e (1.3). Observe que todos os coeficientes presentes em (1.2) são inteiros. Erros de arredondamento são evitados quando se utiliza esta forma (Letchford e Lodi, 2002).

Pré-Multiplicação por um Inteiro

Garfinkel e Nemhauser (1972) discutem a multiplicação da linha do tableau por um inteiro k, antes da geração do corte de Gomory. Dado um critério específico, como maximizar o lado direito, é possível determinar o melhor valor de k.

A proposição abaixo é citada em (Letchford e Lodi, 2002), mas seu resultado não é demonstrado.

Proposição 1.3. Para qualquer inteiro k, o corte

$$\sum_{j \in J} f(k\bar{a}_j) x_j \geq f(k\bar{a}_0) \tag{1.4}$$

é válido para PI. Se $f(\bar{a}_0) < \frac{1}{2}$, k positivo e $\frac{1}{2} \leq kf(\bar{a}_0) < 1$, então o corte (1.4) é no mínimo tão forte quanto o corte fracionário de Gomory (1.3).

Demonstração: Para demonstrar a validade, note que o corte consiste na multiplicação da linha do tableau (1.1) por k e posterior derivação do corte de Gomory (1.3). Para demonstrar a segunda parte da proposição, multiplicamos este corte por k, obtendo

$$\sum_{j \in J} kf(\bar{a}_j) x_j \geq kf(\bar{a}_0), \tag{1.5}$$

e comparamos os coeficientes com o corte em questão. Fazendo $k\bar{a}_j = k(\lfloor \bar{a}_j \rfloor + f(\bar{a}_j))$, temos

$$\begin{aligned} f(k\bar{a}_j) &= f(k \lfloor \bar{a}_j \rfloor + kf(\bar{a}_j)) \\ &= f(kf(\bar{a}_j)) \\ &= kf(\bar{a}_j) - \lfloor kf(\bar{a}_j) \rfloor . \end{aligned}$$

Para o lado direito, temos que $kf(\bar{a}_0) < 1$, por definição. Portanto a equação anterior, para $j = 0$ implica que

$$f(k\bar{a}_0) = kf(\bar{a}_0).$$

Como $\lfloor kf(\bar{a}_j) \rfloor \geq 0$, então

$$f(k\bar{a}_j) \leq kf(\bar{a}_j), \forall\ j \in J.$$

Assim, os coeficientes do lado esquerdo de (1.4) são menores ou iguais aos de (1.5), enquanto o lado direito é igual, e, portanto, o corte (1.4) é no mínimo tão forte quanto o corte (1.5).

Note ainda que como $k \in \mathbb{Z}_+$ e $k > 1$, então $k \geq 2$. Para que o corte gerado seja o mais forte possível, é necessário garantir que a parte fracionária do novo corte seja tal que $f(\bar{a}_0)' = kf(\bar{a}_0) \geq 1/2$, implicando que $\nexists\, k$ tal que o procedimento possa ser reaplicado. □

1.2.2 Corte de Gomory para Programação Inteira Mista

Considere um sistema linear inteiro misto com uma única restrição dado por

$$S = \{(x, y) \in Z_+^n \times R_+^p : \sum_{j=1}^{n} a_j x_j + \sum_{j=1}^{p} g_j y_j = b\}. \tag{1.6}$$

Seja $b = \lfloor b \rfloor + f_0$, tal que $0 < f_0 < 1$ e $a_j = \lfloor a_j \rfloor + f_j$, tal que $0 \leq f_j < 1$. A restrição pode ser reescrita como

$$\sum_{j=1}^{n} f_j x_j + \sum_{j=1}^{p} g_j y_j - f_0 = -\sum_{j=1}^{n} \lfloor a_j \rfloor x_j + \lfloor b \rfloor. \tag{1.7}$$

Considere a partição $\{j : f_j \leq f_0\} \cup \{j : f_j > f_0\}$, e subtraia $\sum\limits_{f_j > f_0} x_j$ de ambos os lados da equação (1.7). Desta forma, temos

$$\sum_{f_j \leq f_0} f_j x_j + \sum_{f_j > f_0} (f_j - 1) x_j + \sum_{j=1}^{p} g_j y_j - f_0 = k\,, \tag{1.8}$$

em que $k = -\sum\limits_{j=1}^{n} \lfloor a_j \rfloor + \lfloor b \rfloor - \sum\limits_{f_j > f_0} x_j$ é inteiro.

Isto implica que o lado esquerdo de (1.8) também é inteiro, e portanto uma das condições a seguir é verdadeira,

$$\sum_{f_j \leq f_0} f_j x_j + \sum_{f_j > f_0} (f_j - 1) x_j + \sum_{j=1}^{p} g_j y_j - f_0 \geq 0, \text{ ou} \tag{1.9}$$

$$\sum_{f_j \leq f_0} f_j x_j + \sum_{f_j > f_0} (f_j - 1) x_j + \sum_{j=1}^{p} g_j y_j - f_0 \leq -1. \tag{1.10}$$

Dividindo (1.9) por f_0 e (1.10) por $-(1 - f_0)$, temos

$$\sum_{f_j \leq f_0} \frac{f_j}{f_0} x_j - \sum_{f_j > f_0} \frac{1 - f_j}{f_0} x_j + \sum_{j=1}^{p} \frac{g_j}{f_0} y_j \geq 1, \text{ ou} \tag{1.11}$$

$$-\sum_{f_j \leq f_0} \frac{f_j}{1 - f_0} x_j + \sum_{f_j > f_0} \frac{1 - f_j}{1 - f_0} x_j - \sum_{j=1}^{p} \frac{g_j}{1 - f_0} y_j \geq 1. \tag{1.12}$$

As expressões acima têm a forma $a^1 z \geq 1$ ou $a^2 z \geq 1$. Isto implica que $\sum_j \max(a_j^1, a_j^2) z_j \geq 1$, para $z \geq 0$. Para cada variável, o coeficiente em (1.11) tem o sinal oposto ao coeficiente em (1.12), e portanto a determinação do coeficiente máximo em (1.11) e (1.12) é trivial.

Portanto, o corte de Gomory (GIM) para PIM é dado por

$$\sum_{f_j \leq f_0} \frac{f_j}{f_0} x_j + \sum_{f_j > f_0} \frac{1-f_j}{1-f_0} x_j + \sum_{g_j > 0} \frac{g_j}{f_0} y_j - \sum_{g_j < 0} \frac{g_j}{1-f_0} y_j \geq 1. \tag{1.13}$$

Observe que o corte GIM foi deduzido a partir de uma restrição genérica. Quando o corte é derivado a partir de uma linha do tableau, as variáveis em (1.13) são não básicas.

No caso de programação inteira pura, o corte acima reduz-se a

$$\sum_{f_j \leq f_0} \frac{f_j}{f_0} x_j + \sum_{f_j > f_0} \frac{1-f_j}{1-f_0} x_j \geq 1, \tag{1.14}$$

denominado corte forte de Gomory (GIF). Como $(1-f_j)/(1-f_0) < f_j/f_0$ quando $f_j > f_0$, fica claro que, de acordo com a Definição 1.10, o corte GIF domina o corte fracionário (GI). Esta mesma observação pode ser utilizada para reduzir GIF a GI. O corte acima pode ser reescrito como um corte mais fraco,

$$\sum_{f_j \leq f_0} \frac{f_j}{f_0} x_j + \sum_{f_j > f_0} \frac{f_j}{f_0} x_j \geq 1, \text{ ou } , \sum_{j \in J} f_j x_j \geq f_0,$$

que corresponde ao corte fracionário (GI) da equação (1.3).

O corte GIM, dado pela desigualdade (1.13), pode ser escrito de outra forma equivalente. Para tanto, considere a i-ésima linha do tableau ótimo da relaxação linear de um problema inteiro misto

$$x_i = \bar{a}_0 + \sum_{j \in J_1} \bar{a}_j (-x_j) + \sum_{j \in J_2} \bar{g}_j (-y_j), \forall\, i \in I, \tag{1.15}$$

em que J_1 é o conjunto das variáveis não básicas inteiras e J_2 é o conjunto das variáveis não básicas não inteiras.

Multiplicando (1.13) por $-f_0$ e somando o resultado a (1.15), chegamos à desigualdade

$$x_i + \sum_{f_j \leq f_0} \lfloor \bar{a}_j \rfloor x_j + \sum_{f_j > f_0} \left(\lfloor \bar{a}_j \rfloor + \frac{f_j - f_0}{1-f_0} \right) x_j + \sum_{\bar{a}_j < 0} \frac{\bar{a}_j}{1-f_0} y_j \leq \lfloor \bar{a}_0 \rfloor . \tag{1.16}$$

A desigualdade acima fornece uma maneira alternativa de escrever o corte misto em termos de variáveis básicas e não básicas, e tem um número menor de coeficientes fracionários que a sua forma equivalente (1.13). A presença de coeficientes inteiros evita erros de arredondamento (Letchford e Lodi, 2002). Portanto, para uma eventual implementação, esta forma deve ser priorizada.

Este fato também é válido para o corte GIF. Neste caso, a equação acima, sem variáveis reais, torna-se

$$x_i + \sum_{f_j \leq f_0} \lfloor \bar{a}_j \rfloor x_j + \sum_{f_j > f_0} \left(\lfloor \bar{a}_j \rfloor + \frac{f_j - f_0}{1-f_0} \right) x_j \leq \lfloor \bar{a}_0 \rfloor . \tag{1.17}$$

A variável de folga de ambos os cortes não é necessariamente inteira. Adicionando a variável de folga, s, à equação (1.16), ela pode ser reescrita como

$$s = \lfloor \bar{a}_0 \rfloor - (x_i + \sum_{f_j \leq f_0} \lfloor \bar{a}_j \rfloor \, x_j) - \sum_{f_j > f_0} \left(\lfloor \bar{a}_j \rfloor + \frac{f_j - f_0}{1 - f_0} \right) x_j - \sum_{\bar{a}_j < 0} \frac{\bar{a}_j}{1 - f_0} y_j.$$

Portanto, tanto o corte GIM quanto o corte GIF geram variáveis de folga reais quando inseridos em um PL. Este é um fato que afeta diretamente a implementação de cortes para problemas apenas com variáveis inteiras. Considere, como exemplo, um PI em que decida-se inserir cortes do tipo GIF. A inserção destes cortes requer a definição de novas variáveis de folga, de forma a definir a igualdade do tableau. Suponha um corte GIF é inserido na relaxação linear de um PI com variáveis $\{x_1, ..., x_n\} \in Z_+$. É feita a reotimização pelo método dual simplex e o processo é repetido: é inserida novamente outra desigualdade do tipo GIF. Note que o conjunto de variáveis $\{x_1, ..., x_n, s\} \notin Z_+$, pois $s \in R_+$. Sendo assim, deve-se adotar um procedimento de remoção da variável s antes da geração do corte GIF, ou aplicar o corte GIM. A remoção da variável s consiste em expressá-la como função das variáveis $\{x_1, ..., x_n\}$.

Validade global de GIM

Balas et al. (1996b) demonstram como tornar cortes de Gomory gerados em um determinado nó da árvore de *branch-and-bound* válidos para qualquer outro nó. A validade global dos cortes é demonstrada para PIM com variáveis 0-1, e se baseia em um procedimento de *lifting*.

Considere a i-ésima linha do tableau ótimo da relaxação linear de um problema inteiro misto apenas com variáveis 0-1,

$$x_i = \bar{a}_0 + \sum_{j \in J_1} \bar{a}_j(-x_j) + \sum_{j \in J_2} \bar{g}_j(-y_j), \forall \, i \in I,$$

em que J_1 é o conjunto das variáveis não básicas binárias e J_2 é o conjunto das variáveis não básicas reais.

Sejam F_0 e F_1, os conjuntos de variáveis binárias não básicas fixas em 0 e em 1, respectivamente, em um determinado nó da árvore. Um corte gerado neste nó só é válido em outro nó em que as variáveis em $F_0 \cup F_1$ permanecem fixas. A próxima proposição demonstra como é possível tornar válido globalmente o corte GIM.

Proposição 1.4. Seja a i-ésima linha do PL em um determinado nó da árvore de *branch-and-bound* dada por

$$\begin{aligned}
& x_i = \bar{a}_0 + \sum_{j \in J_1} \bar{a}_j(-x_j) + \sum_{j \in J_2} \bar{g}_j(-y_j), \forall \, i \in I, \\
& x_k, y_k \geq 0 \quad \forall \, k \in I \cup J, \\
& x_k \leq 0 \quad \forall \, k \in F_0, \\
& x_k \geq 1 \quad \forall \, k \in F_1,
\end{aligned}$$

em que F_0 e F_1 são as variáveis binárias não básicas fixas em 0 e em 1. Aplicando-se o *lifting* das variáveis em F_1, o corte GIM (1.13) é valido globalmente.

Demonstração: Aplica-se *lifting* a todas as variáveis em F_1. Ou seja, substituem-se todas as variáveis x_k em F_1 por $1 - x_k$, de forma a obter um novo problema em que estas variáveis estão fixas em 0. Portanto $F_1 = \emptyset$.

Suponha, a princípio, que $F_0 = \emptyset$. O corte GIM é derivado na seção anterior aplicando-se o procedimento que obtém a equação (1.13) a partir da (1.6). Demonstra-se que o procedimento continua válido se $F_0 \neq \emptyset$.

A equação (1.8) permanece válida para $F_0 \neq \emptyset$, pois $-\sum_{j=1}^{n} \lfloor a_j \rfloor + \lfloor b \rfloor - \sum_{f_j > f_0} x_j$ permanece inteiro, e, portanto, as desigualdades (1.9) e (1.10) continuam válidas. A desigualdade do corte (1.13) é derivada obtendo-se o máximo entre os coeficientes das variáveis x_i e y_i em (1.11) e (1.12). O resultado do procedimento é o mesmo, se $F_0 \neq 0$. Assim, a desigualdade (1.13) é válida se um subconjunto qualquer de variáveis binárias estiverem fixas em 0. Portanto, o corte é válido para qualquer nó da árvore de *branch-and-cut*. □

Exemplo 1.3. Considere o seguinte problema inteiro misto

$$\begin{array}{ll} \max & 3x_1 + 6x_2 + x_3 + x_4 \\ \text{s.a.} & 2x_1 + 3x_2 + x_3 + x_4 = 4, \\ & x_i \in \{0,1\}, i = 1, ..., 3, \\ & x_4 \geq 0. \end{array}$$

A solução ótima da relaxação linear é $x_1 = 1/2$, $x_2 = 1$, $x_3 = x_4 = 0$. Considere que a variável x_1 é escolhida para efetuar ramificação. Se fixarmos $x_1 = 1$, ou seja, $F_1 = \{1\}$, o novo nó é dado por

$$\begin{array}{ll} \max & 6x_2 + x_3 + x_4 \\ \text{s.a.} & x_2 + 1/3x_3 + 1/3x_4 = 2/3, \\ & 0 \leq x_2, x_3 \leq 1, \\ & x_4 \geq 0, \end{array} \tag{1.18}$$

cuja solução ótima é $x_1 = 1$, $x_2 = 2/3$, $x_3 = x_4 = 0$. O corte GIM calculado para a linha deste tableau é

$$x_3 + x_4 \geq 2.$$

Note que este corte não é válido para o nó raiz, pois não satisfaz a todas as soluções factíveis do problema inicial. Por exemplo, o ponto $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$ é válido para o nó raiz e é excluído por este corte no novo nó.

Para que o corte seja válido globalmente, as variáveis são fixadas em 0 ou 1 através da alteração de seu limitante superior e inferior, respectivamente e estas variáveis devem permanecer na formulação em todos os nós da árvore. Portanto, a variável x_1 deve ser mantida na formulação e deve ser aplicado o procedimento de *lifting* desta variável. Assim, se mantivermos x_1, o novo PL é dado por

$$\begin{array}{ll} \max & 3x_1 + 6x_2 + x_3 + x_4 \\ \text{s.a.} & 2x_1 + 3x_2 + x_3 + x_4 = 4, \end{array}$$

$$0 \leq x_2, x_3 \leq 1,$$
$$1 \leq x_1 \leq 1,$$
$$x_4 \geq 0.$$

Seja $x'_1 = 1 - x_1$, de forma que a nova variável está fixa em 0. Assim, a nova linha do tableau é dada por

$$x_2 - \frac{2}{3}x'_1 + \frac{1}{3}x_3 + \frac{1}{3}x_4 = \frac{2}{3},$$

cujos coeficientes são iguais aos da linha do tableau (1.18), a não ser pelo coeficiente de x'_1. Assim, o novo corte GIM é

$$x'_1 + x_3 + x_4 \geq 2,$$

que pode ser colocado em função da variável x_1, originado o corte válido para o nó raiz e qualquer outro nó da árvore,

$$-x_1 + x_3 + x_4 \geq 1. \quad \square$$

1.3 Cortes Chvátal-Gomory

Em um artigo marcante, Chvátal (1973) propôs uma forma de construção de desigualdades válidas para o conjunto $X = P \cap Z^n$, em que $P = \{x \in R^n : Ax \leq b\}$ é um poliedro em que os elementos de A e b são racionais, A é uma matriz $m \times n$, com colunas $\{a^1, a^2, \ldots, a^n\}$, e u é um vetor linha tal que $u \in R^m_+$. Essa forma de construção é denominada procedimento de Chvátal-Gomory (Nemhauser e Wolsey, 1988) e contém três passos:

(i) A desigualdade

$$\sum_{j=1}^{n} ua^j x_j \leq ub$$

é válida para P, pois $u \geq 0$ e $\sum_{j=1}^{n} a^j x_j \leq b$;

(ii) A desigualdade

$$\sum_{j=1}^{n} \lfloor ua^j \rfloor x_j \leq ub$$

é válida para P, pois $x \geq 0$;

(iii) A desigualdade

$$\sum_{j=1}^{n} \lfloor ua^j \rfloor x_j \leq \lfloor ub \rfloor$$

é válida para X, pois x é inteiro, e portanto $\sum_{j=1}^{n} \lfloor ua^j \rfloor x_j$ é inteiro.

A desigualdade

$$\sum_{j=1}^{n} \lfloor ua^j \rfloor x_j \leq \lfloor ub \rfloor, \tag{1.19}$$

ou $\lfloor uA \rfloor x \leq \lfloor ub \rfloor$, chamada desigualdade Chvátal-Gomory (CG), pode ser adicionada ao sistema $Ax \leq b$, e o procedimento pode ser repetido. O teorema a seguir é devido a Chvátal (1973).

Teorema 1.1. Toda desigualdade válida para X pode ser obtida pela aplicação do procedimento de Chvátal-Gomory por um número finito de vezes. □

O número de aplicações do procedimento CG para obter qualquer desigualdade válida é denominado *rank* (Chvátal, 1973), como explicitado na definição a seguir.

Definição 1.11. Uma desigualdade $\pi x \leq \pi_0$ para X é de *rank* k com relação à $P \subseteq R^n_+$ se $\pi x \leq \pi_0$ não é equivalente ou dominada por qualquer combinação linear de desigualdades CG, cada desigualdade determinada pela aplicação de no máximo $k-1$ aplicações do procedimento CG, mas é equivalente ou dominada por uma combinação linear de desigualdades CG, cada desigualdade determinada pela aplicação de no máximo k aplicações do procedimento CG. □

O teorema anterior mostra que, para um poliedro racional, toda desigualdade válida para X é de *rank* finito.

A proposição a seguir, correspondente ao teorema 6.34 em (Cook et al., 1998), mostra que os elementos do vetor u da desigualdade CG podem ser limitados entre os valores 0 e 1.

Proposição 1.5. Qualquer desigualdade CG pode ser escrita como $\lfloor uA \rfloor x \leq \lfloor ub \rfloor$, em que $0 \leq u_i < 1, i = 1, \ldots, m$.

Demonstração: Como o poliedro $P = \{x : Ax \leq b\}$ é racional, pode-se assumir, sem perda de generalidade, que os elementos de A e b são inteiros. Seja $wx \leq t$ uma desigualdade CG, em que $w = \lfloor \bar{u}A \rfloor, t = \lfloor \bar{u}b \rfloor$. Seja $\bar{u}' = \bar{u} - \lfloor \bar{u} \rfloor$ a parte fracionária de $\bar{u}$. Então, $w' = \lfloor \bar{u}'A \rfloor = w - \lfloor \bar{u} \rfloor A$ é um vetor inteiro, e $t' = \lfloor \bar{u}'b \rfloor = t - \lfloor \bar{u} \rfloor b$ difere de t por um valor inteiro. Portanto, a adição da desigualdade CG $w'x \leq t'$ à desigualdade CG $\lfloor \bar{u} \rfloor Ax \leq \lfloor \bar{u} \rfloor b$ (uma combinação linear não negativa de $Ax \leq b$) gera a desigualdade CG $wx \leq t$. Isto implica que qualquer desigualdade distinta daquelas geradas por $\lfloor uA \rfloor x \leq \lfloor ub \rfloor$, em que $0 \leq u_i < 1, i = 1, \ldots, m$, é redundante. □

1.3.1 Equivalência entre o Corte de Chvátal-Gomory e o Corte Fracionário de Gomory

Cornuéjols e Li (2001) comparam diversas famílias de cortes propostas na literatura através do conceito de *fecho elementar*, associado com todos os cortes da família. Este conceito foi introduzido por Chvátal (1973), e é definido a seguir.

Definição 1.12. Seja $X = P \cap (Z^n_+ \times R^p_+)$, e uma família F de desigualdades $\pi x + \gamma y$ geradas de $P = \{(x, y) \in R^{n+p}_+ : Ax + Gy \leq b\}$ e válidas para X. O *fecho elementar* P_F é um conjunto convexo obtido pela interseção de todas as desigualdades de F. □

A seguir, compara-se o fecho elementar F_{CG} da família de cortes CG com o fecho elementar F_G da família de cortes de fracionários de Gomory.

Considere o poliedro $P = \{x \in R^n_+ : Ax \leq b\}$ e assuma, sem perda de generalidade, que os elementos de A e b são inteiros. Note que P é equivalente a $P^{'} = \{(x, s) \in R^{n+m}_+ : Ax + s = b\}$.

Seja $P^{'}_I = \{(x, s) \in Z^{n+m}_+ : Ax + s = b\}$. Para o vetor $u \in R^m$, a seguinte desigualdade pode ser obtida:

$$u[A \quad I] \begin{bmatrix} x \\ s \end{bmatrix} - \lfloor u[A \quad I] \rfloor \begin{bmatrix} x \\ s \end{bmatrix} \geq ub - \lfloor ub \rfloor.$$

Substituindo $s = b - Ax$ nesta desigualdade, obtém-se o corte fracionário $\lfloor uA \rfloor x - \lfloor u \rfloor Ax \leq \lfloor ub \rfloor - \lfloor u \rfloor b$ no espaço das variáveis x.

A proposição a seguir é devida a Cornuéjols e Li (2001).

Proposição 1.6. $P_{CG} = P_F$.

Demonstração: Seja $Ax \leq b, x \in Z^n_+$. Pela desigualdade (1.19), qualquer corte CG tem a forma $\lfloor uA \rfloor x \leq \lfloor b \rfloor$, tal que $0 \leq u_i < 1$, para $i = 1, \ldots, m$. Como $\lfloor u \rfloor = 0$, obtém-se então o corte fracionário de Gomory $\lfloor uA \rfloor x - \lfloor u \rfloor Ax \leq \lfloor ub \rfloor - \lfloor u \rfloor b$. Por outro lado, qualquer corte fracionário de Gomory $\lfloor uA \rfloor x - \lfloor u \rfloor Ax \leq \lfloor ub \rfloor - \lfloor u \rfloor b$ pode ser escrito como $\lfloor uA - \lfloor u \rfloor A \rfloor x \leq \lfloor ub - \lfloor u \rfloor b \rfloor$, pois os elementos de A e b são inteiros. Esta desigualdade é um corte CG $\lfloor \lambda A \rfloor \leq \lfloor \lambda b \rfloor$, tal que $\lambda = u - \lfloor u \rfloor$. □

1.4 Algoritmo *branch-and-cut*

Métodos *branch-and-cut* são algoritmos exatos que combinam algoritmos de planos de corte com o método *branch-and-bound*. Este último consiste na sucessiva resolução de relaxações lineares que constituem sub-problemas do PIM inicial.

O Algoritmo 1 a seguir descreve o método *branch-and-cut*. O algoritmo é aplicado sobre um PIM de maximização cuja formulação é P. Assim, temos $z = \max\{cx : x \in X\}$, em que X é o conjunto de pontos factíveis dado por $X = P \cap (Z^n_+ \times R^p_+)$.

Uma lista L de nós ativos mantém os nós da árvore ainda não processados. Quando o i-ésimo nó, correspondente a P^i, é removido desta lista, a relaxação linear correspondente é resolvida. Se for infactível, o nó é eliminado e o processo reinicia-se com o próximo nó de L. Caso contrário pode haver ou não a adição de cortes. A inserção de cortes em P^i na iteração k, gera P^i_k. É feita reotimização do PL e o contador de iterações k é incrementado. O processo se repete até que o problema se torne infactível ou opte-se por não inserir mais cortes.

Se o nó não é infactível são aplicados os critérios padrões de eliminação de nó do método *branch-and-bound*. Se $\bar{z}^{i,k} \leq \underline{z}$, ou seja, o limitante superior do nó atual é menor ou igual à melhor solução inteira encontrada, então este nó pode ser eliminado. Se a solução for inteira, o limitante inferior global e a solução incumbente são atualizados. Se a solução não for inteira é feita ramificação.

O procedimento de ramificação, ou *branching*, consiste em particionar o problema atual em dois ou mais problemas. Em geral a partição é feita criando-se dois novos problemas em que são adicionadas restrições sobre uma variável inteira com valor fracionário. O próximo exemplo ilustra a aplicação do método *branch-and-cut*.

Algoritmo 1: Algoritmo *branch-and-cut*

1 **Início:** Seja P uma formulação para o problema $z = \max\{cx : x \in X\}$, em que $X = P \cap (Z_+^n \times R_+^p)$. Seja L o conjunto de nós ativos na árvore de *branch-and-cut*. Resolva o problema inicial e insira-o em L. Faça $\underline{z} = -\infty$ e a solução incumbente $x^* = \emptyset$.

2 **Seleção de Nó:** Se $L = \emptyset$, então vá para **Término**. Senão, selecione e remova o nó i de L. Seja P^i a formulação relativa ao conjunto X^i deste nó. Faça $k = 1$ e $P^{i,1} = P^i$.

3 **Relaxação do PL atual:** Resolva $\bar{z}^{i,k} = \max\{cx : x \in P^{i,k}\}$. Se o problema for infactível, elimine o nó atual e vá para **Seleção de Nó**. Senão, faça $x^{i,k}$ a solução atual.

4 **Adição de cortes:** Se optar-se pela não geração de cortes ou se não for possível gerá-los, vá para **Eliminação**. Caso contrário, adicione cortes a $P^{i,k}$, gerando $P^{i,k+1}$. Incremente o valor de k e volte ao passo **Relaxação do PL atual**.

5 **Eliminação:** Se $\bar{z}^{i,k} \leq \underline{z}$ vá para **Seleção de Nó**. Se $x \in X$, faça $\underline{z} = \bar{z}^{i,k}$, atualize solução incumbente, $x^* = x^{i,k}$ e vá para **Seleção de Nó**. Caso contrário, vá para **Ramificação**.

6 **Ramificação:** Crie dois ou mais novos problemas X_t^i com formulações P_t^i. Adicione-os à lista L e vá para **Seleção de Nó**.

7 **Término:** Retorne a solução incumbente x^* e seu valor ótimo correspondente, $\underline{z}$.

Exemplo 1.4. Considere o PI de maximização a ser resolvido pelo método *branch-and-cut*,

$$\begin{array}{ll} \max & x_1 + 2x_2 \\ \text{s.a.} & 3x_1 + x_2 \leq 11 \\ & -x_1 + 2x_2 \leq 5 \\ & x \in Z_+^2. \end{array}$$

A resolução deste problema através do método *branch-and-cut* está representada na Figura 1.6. Adicionando-se variáveis de folga, $s_1 \in Z_+$ e $s_2 \in Z_+$, e resolvendo a relaxação linear correspondente ao nó raiz, nó 0, obtém-se o tableau correspondente à solução ótima,

$$\begin{array}{ll} \max & \dfrac{69}{7} - \dfrac{4}{7}s_1 - \dfrac{5}{7}s_2 \\ \text{s.a.} & x_1 + \dfrac{2}{7}s_1 - \dfrac{1}{7}s_2 = \dfrac{17}{7} \\ & x_2 + \dfrac{1}{7}s_1 + \dfrac{3}{7}s_2 = \dfrac{26}{7} \\ & (x, s) \in R_+^4. \end{array}$$

A solução ótima é o ponto fracionário $(x_1, x_2) = (\frac{17}{7}, \frac{26}{7})$. A partir da primeira igualdade do tableau acima é gerado o corte forte de Gomory (GIF). De acordo com a desigualdade (1.14), este corte é dado por $8s_1 + 3s_2 \geq 12$, e pode ser colocado em função das variáveis originais, gerando $3x_1 + 2x_2 \leq 13$. Adicionando esta desigualdade ao PL anterior e reotimizando, obtém-se o novo

tableau ótimo

$$\begin{aligned}
\max \quad & 9 - \frac{1}{2}s_2 - \frac{1}{2}s_3 \\
\text{s.a.} \quad & x_1 - \frac{1}{4}s_2 + \frac{1}{4}s_3 = 2 \\
& x_2 + \frac{3}{8}s_2 + \frac{1}{8}s_3 = \frac{7}{2} \\
& s_1 + \frac{3}{8}s_2 - \frac{7}{8}s_3 = \frac{3}{2} \\
& (x, s) \in R^5_+,
\end{aligned}$$

em que $s_3 \in Z_+$ é a variável de folga associada ao corte GIF.

Note que a solução ótima $(x_1, x_2) = (2, \frac{7}{2})$ continua fracionária em x_2. Esta variável é escolhida para ramificação, gerando os nós 1 e 2. Ao PL do nó 1 é adicionada a restrição $x_2 \geq 4$ e ao nó 2 é adicionada a restrição $x_2 \leq 3$. Como mostra a Figura 1.7, a região factível do problema do nó 1 é vazia, e, portanto, este nó é eliminado por infactibilidade.

A solução ótima do nó 2, adicionada a variável de folga $s_4 \in Z_+$ é dada por

$$\begin{aligned}
\max \quad & 25/3 - \frac{1}{3}s_3 - \frac{4}{3}s_4 \\
\text{s.a.} \quad & x_1 + \frac{1}{3}s_3 - \frac{2}{3}s_4 = \frac{7}{3} \\
& x_2 + s_4 = 3 \\
& s_2 + \frac{1}{3}s_3 - \frac{8}{3}s_4 = \frac{4}{3} \\
& s_1 - s_3 + s_4 = 1 \\
& (x, s) \in R^6_+ .
\end{aligned}$$

O corte fracionário gerado a partir da primeira equação do tableau é $x_1 - s_4 \leq 2$, e pode ser colocada em função das variáveis originais, gerando a faceta $x_1 + x_2 \leq 5$. Como mostra a Figura 1.7, a adição desta desigualdade ao nó 2, seguida de reotimização, dá origem ao ponto ótimo inteiro $(2, 3)$, e o nó é eliminado por otimalidade. □

1.5 Medidas de Qualidade de Cortes

Em um algoritmo de plano de cortes, o número de cortes disponíveis para inserção pode ser excessivo. A inserção de novos cortes em um nó leva a um tempo maior de solução da relaxação linear correspondente. Se estes cortes não são bem selecionados, o tempo total de resolução do nó pode exceder o benefício gerado pela inserção do corte.

Este fato cria a necessidade de definir critérios de classificação dos cortes que estão disponíveis para inserção, de forma a dar prioridade para os cortes mais fortes. A literatura de cortes cita duas medidas bastante utilizadas. Ambas as medidas são usadas no contexto do algoritmo de *branch-and-cut* por Balas et al. (1996a,b) e procuram quantificar a noção de profundidade de um corte.

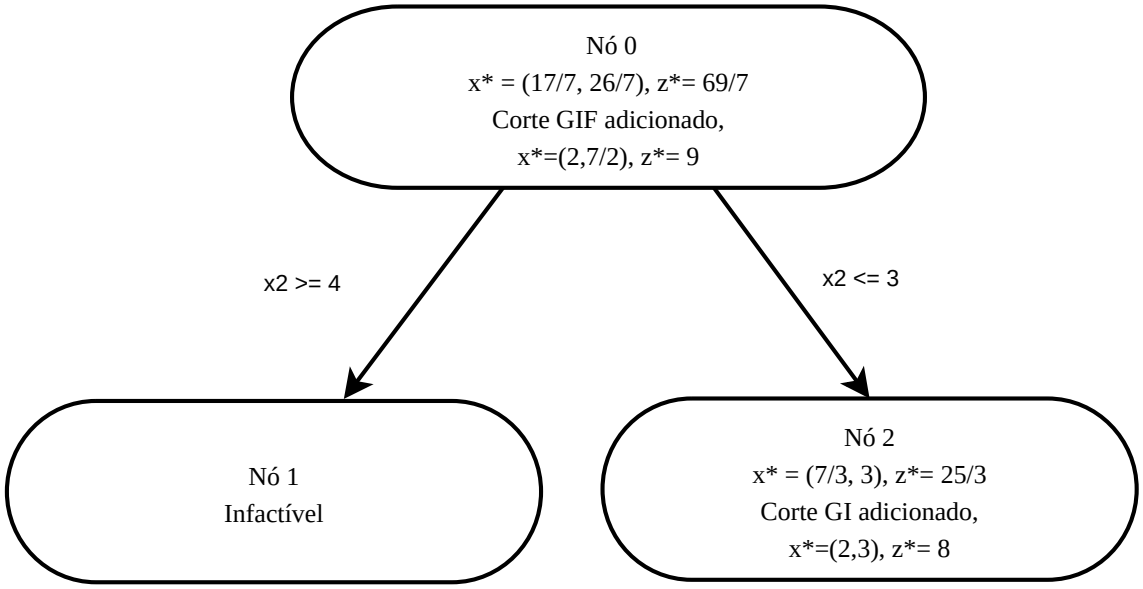

Figura 1.6: Árvore gerada pela aplicação do método *branch-and-cut* ao exemplo 1.4

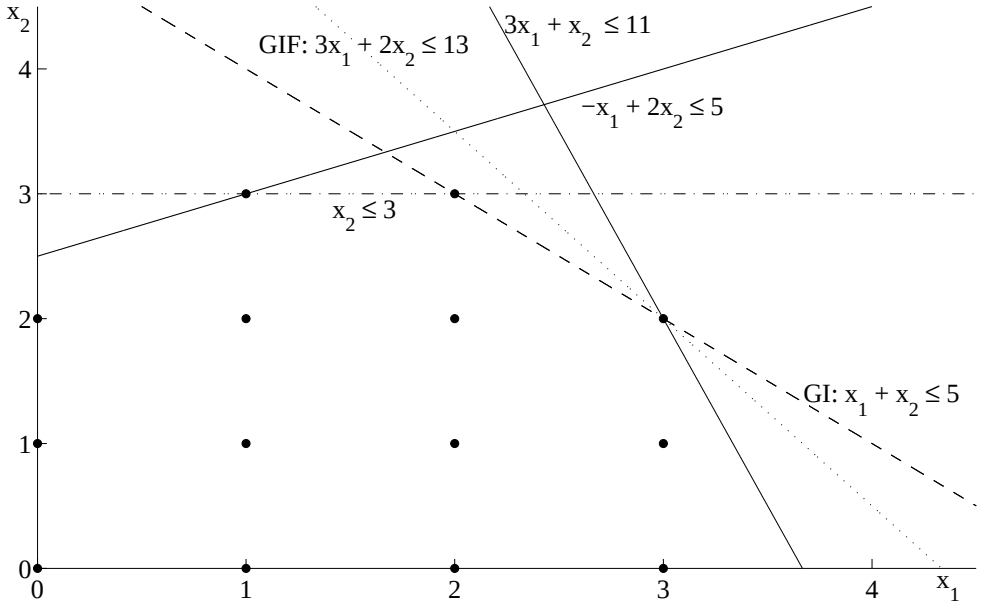

Figura 1.7: Aplicação de *branch-and-cut* ao exemplo 1.4

A primeira delas é a *violação* do corte em relação à solução do nó. Dado um corte $\alpha x \leq \alpha_0$, e um ponto x^*, a violação deste corte em relação a este ponto é dada por $\alpha x^* - \alpha_0$. Se a violação for positiva, o corte elimina o ponto x^*.

A normalização desta medida foi proposta em (Balas et al., 1996a), de duas maneiras diferentes. Para cortes com lado direito diferente de zero, esta normalização pode ser relativa ao lado direito do

corte, ou seja, todos os coeficientes são divididos por α_0. O outro tipo de normalização é relativo aos coeficientes do lado esquerdo e consiste em dividir a desigualdade do corte pela soma dos coeficientes do lado esquerdo (em módulo) gerando um corte em que a soma dos coeficientes em módulo é igual a 1.

Definição 1.13. Dado um corte $\alpha x \leq \alpha_0$, a correspondente *violação normalizada* em relação ao ponto x^* será dada por

$$\text{viol}(x^*, \alpha, \alpha_0) = \frac{\sum_i \alpha_i x_i^* - \alpha_0}{\sum_i |\alpha_i|}, \tag{1.20}$$

e um corte *viola* a solução x^* se $\text{viol}(x^*, \alpha, \alpha_0) > 0$. □

A segunda medida é a distância euclidiana entre a solução ótima do nó atual e o plano de corte. Andreello et al. (2007) propõem novamente o uso da distância euclidiana como medida de eficiência dos cortes.

Definição 1.14. Dado um corte $\alpha x \leq \alpha_0$, e um ponto x^*, a *distância euclidiana* entre o plano de corte dado por $\alpha x = \alpha_0$ e o ponto x^* é dada por

$$\text{dist}(x^*, \alpha, \alpha_0) = \frac{|\alpha x^* - \alpha_0|}{\|\alpha\|}, \tag{1.21}$$

em que $\|\alpha\|$ é a norma euclidiana do vetor α. □

Como já era apontado em (Balas et al., 1996a), a presença de componentes nulas na solução x^* pode distorcer esta medida. Suponha, como exemplo, um corte com três componentes, $\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 \leq \alpha_0$, em que $\alpha_i > 0$ e solução ótima $x_1^* > 0$, $x_2^* > 0$ e $x_3^* = 0$. A distância euclidiana segundo a equação (1.21) é dada por

$$\frac{|\alpha_1 x_1^* + \alpha_2 x_2^* - \alpha_0|}{\sqrt{\alpha_1^2 + \alpha_2^2 + \alpha_3^2}}.$$

Se considerarmos agora o mesmo corte, mas com $\alpha_3 = 0$, temos $\alpha_1 x_1 + \alpha_2 x_2 \leq \alpha_0$, que é claramente dominado pelo corte anterior. Entretanto, se calcularmos sua distância euclidiana em relação à mesma solução, temos

$$\frac{|\alpha_1 x_1^* + \alpha_2 x_2^* - \alpha_0|}{\sqrt{\alpha_1^2 + \alpha_2^2}},$$

que denota que este novo corte seria mais profundo que o corte anterior. Esta é uma característica de distorção inerente a esta medida de qualidade. Ao mesmo tempo, é uma medida que pode ser usada como guia na comparação de cortes. Desta forma, define-se a eficiência de um corte como a distância euclidiana, sem que seja tomado o módulo. Assim, uma eficiência negativa indica que não há violação da desigualdade em relação à solução atual e pode ser definida uma eficiência mínima para que o corte seja utilizado.

Definição 1.15. Dado um corte $\alpha x \leq \alpha_0$, e um ponto x^*, a *eficiência* com que o plano de corte dado por $\alpha x = \alpha_0$ elimina o ponto x^* é dada por

$$\text{efic}(x^*, \alpha, \alpha_0) = \frac{\alpha^T x^* - \alpha_0}{\|\alpha\|}, \tag{1.22}$$

em que $\|\alpha\|$ é a norma euclidiana do vetor α. □

Outro aspecto importante em algoritmos de planos de corte é a definição de uma medida de semelhança entre cortes. Em (Balas et al., 1996a), os autores propõem o descarte de cortes duplicados gerados no mesmo *round* (um *round* de cortes é obtido gerando-se um corte para cada linha do tableau com lado direito fracionário) através da medida do cosseno do ângulo entre os vetores normais a dois planos de corte. Se chamarmos de θ o ângulo entre estes dois vetores, eles serão paralelos quando $\theta = 0$, e , portanto, $cos(\theta) = 1$. No trabalho citado, um corte é adicionado ao PL se $cos(\theta) \leq 0.999$ entre o novo corte e todos aqueles já adicionados.

Definição 1.16. O paralelismo entre dois cortes, $\alpha x \leq \alpha_0$ e $\beta x \leq \beta_0$, é a medida do cosseno do ângulo entre os dois vetores normais a cada plano e é dado por

$$\text{par}(\alpha, \beta) = \frac{\left|\alpha^T \beta\right|}{\|\alpha\| \, \|\beta\|}, \tag{1.23}$$

de forma que dois vetores paralelos têm par(α, β) = 1. □

Esta medida de paralelismo entre dois cortes também foi utilizada em (Andreello et al., 2007). Neste trabalho os autores propõem a definição de um parâmetro $max_{par} \in [0, 1)$, que é o máximo valor para $cos(\theta)$. Um corte $\alpha x \leq \alpha_0$ é descartado quando tem uma eficiência menor que outro corte $\beta x \leq \beta_0$, tal que par $(\alpha, \beta) > max_{par}$.

Capítulo 2

Descrição dos Cortes Implementados

Neste capítulo são apresentados os três planos de corte para programação inteira implementados neste trabalho: Cortes Fracionários Fortes propostos por Letchford e Lodi (2002), K-cortes, propostos por Cornuéjols et al. (2003) e cortes Chvátal-Gomory-Nível, propostos por Glover e Sherali (2005). Para cada corte é apresentada a sua derivação, análise de parâmetros e resultados obtidos pelos seus autores.

2.1 Cortes Genéricos em Programação Inteira

Desigualdades válidas podem ser divididas entre aquelas desenvolvidas para problemas específicos e que utilizam, portanto, informações a respeito da estrutura destes problemas, e aquelas que são genéricas, ou seja, são derivadas por técnicas algébricas aplicáveis a quaisquer problemas de programação inteira ou programação inteira mista.

O uso de desigualdades válidas genéricas para resolução de problemas de programação inteira teve início com os cortes fracionários de Gomory (1958), quando introduziu o método de formas inteiras (*The Method of Integer Forms*).

Chvátal (1973) introduziu os cortes posteriormente denominados Chvátal-Gomory, por serem equivalentes aos cortes fracionários. Os cortes disjuntivos foram introduzidos por Balas (1979). Nemhauser e Wolsey (1990) propuseram o corte *mixed integer rounding* (MIR) que é motivado pelo corte de Gomory para programação inteira mista. Os cortes *lift-and-project* para problemas inteiros mistos 0-1 foram propostos por Balas et al. (1993).

Padberg e Rinaldi (1991) introduziram o método *branch-and-cut*, para resolução do problema do caixeiro viajante. Balas et al. (1996b) demonstraram a validade global dos cortes de Gomory para problemas inteiros mistos 0-1 e obtiveram resultados satisfatórios com a adição de mais de um corte aos nós da árvore de *branch-and-cut*.

Letchford e Lodi (2002) apresentam uma nova classe de cortes derivados dos cortes fracionários de Gomory e do corte de Chvátal-Gomory, denominados Corte Fracionário Forte (CFF) e corte Chvátal-Gomory Forte (CGF) que dominam os cortes originais. A derivação destes cortes é apresentada na Seção 2.2, juntamente com resultados computacionais obtidos pelos autores, exemplo de aplicação e algoritmo usado para testes.

Os K-cortes foram propostos por Cornuéjols et al. (2003) e consistem em uma variação do corte

de Gomory para programação inteira mista aplicado a problemas de programação inteira. As linhas do tableau são multiplicadas por um inteiro k antes da geração do corte. A Seção 2.3 apresenta o procedimento de derivação deste corte, resultados computacionais obtidos pelos autores, exemplo de aplicação e algoritmo para determinação do parâmetro k.

Glover e Sherali (2005) propuseram uma nova classe de cortes derivados do corte de Chvátal-Gomory, denominada CG-Nível (*CG-Tier*). O corte consiste na determinação de dois parâmetros. O nível do corte, p, denota o número de vezes que o procedimento básico de derivação do corte é aplicado, enquanto que o parâmetro d é usado na divisão dos coeficientes da equação original. A derivação deste corte, análise de parâmetros e exemplos estão na Seção 2.4.

2.2 Corte Chvátal-Gomory Forte e Corte Fracionário Forte

Seja $X = P \cap Z^n$, em que $P = \{x \in R^n : Ax = b\}$. Como visto na Seção 1.3, o corte de Chvátal-Gomory é calculado sobre a combinação linear das equações de P. Seja $N = \{1, ..., n\}$ e a equação resultante desta combinação dada por

$$\sum_{i \in N} a_i x_i = a_0. \tag{2.1}$$

A desigualdade de Chvátal-Gomory é dada por (1.19) e pode ser reescrita como

$$\sum_{i \in N} \lfloor a_i \rfloor x_i \leq \lfloor a_0 \rfloor . \tag{2.2}$$

Para o teorema a seguir define-se a notação $a_i = \lfloor a_i \rfloor + f_i$.

Teorema 2.1. Considere a igualdade (2.1) em que $f_0 > 0$, e seja $k \geq 1$ o único inteiro tal que

$$1/(k+1) \leq f_0 < 1/k. \tag{2.3}$$

Seja *N* particionado em $k+1$ conjuntos da seguinte forma:

$$N_0 = \left\{ i \in N : f_i \leq f_0 \right\}, \text{ e, para } p = 1, \ldots, k,$$
$$N_p = \left\{ i \in N : f_0 + \frac{(p-1)(1-f_0)}{k} < f_i \leq f_0 + \frac{p(1-f_0)}{k} \right\}.$$

O corte Chvátal-Gomory Forte (CGF) é definido como a desigualdade

$$\sum_{i \in N_0} (k+1) \lfloor a_i \rfloor x_i + \sum_{p=1}^{k} \sum_{i \in N_p} \left((k+1) \lfloor a_i \rfloor + p \right) x_i \leq (k+1) \lfloor a_0 \rfloor , \tag{2.4}$$

e domina o corte clássico de Chvátal-Gomory (2.2).

Demonstração: Multiplicando (2.1) por k, o corte de Chvátal-Gomory resulta

$$\sum_{i \in N} \lfloor k a_i \rfloor \, x_i \leq \lfloor k a_0 \rfloor \, .$$

Dado que $\lfloor k a_i \rfloor = k \lfloor a_i \rfloor + \lfloor k f_i \rfloor$, podemos reescrever a inequação anterior como

$$\sum_{i \in N} \left(k \lfloor a_i \rfloor + \lfloor k f_i \rfloor \right) x_i \leq k \lfloor a_0 \rfloor + \lfloor k f_0 \rfloor \, . \tag{2.5}$$

Por definição, $k f_0 < 1$, e então, $\lfloor k f_0 \rfloor = 0$. Define-se o inteiro p tal que

$$p \leq k f_i < (p+1), \tag{2.6}$$

e portanto, $\lfloor k f_i \rfloor = p$. Como $0 \leq f_i < 1$, então $0 \leq p \leq k-1$ e, assim, a inequação (2.5) pode ser reescrita como

$$\sum_{p=0}^{k-1} \sum_{i \in N : p/k \leq f_i < (p+1)/k} \left(k \lfloor a_i \rfloor + p \right) x_i \leq k \lfloor a_0 \rfloor \, . \tag{2.7}$$

Seja $\epsilon > 0$ um número real pequeno e tal que

$$1 - \epsilon \geq f_0 / f_i, \quad \forall \, i \in N \backslash N_0.$$

Para obter o corte é feita a combinação linear da equação (2.1) e da desigualdade (2.7) utilizando dois multiplicadores positivos, de forma a garantir a validade da desigualdade originada. Seja $M_1 = (1-\epsilon)/f_0$. Note que $M_1 > 0$, pois $1 - \epsilon > 0$ e $f_0 > 0$. O segundo multiplicador, M_2 é definido como

$$M_2 = 1 + \frac{1}{k} \left(1 - \frac{1-\epsilon}{f_0} \right) \geq 1 + \frac{1}{k} \left(1 - \frac{1}{f_0} \right) .$$

Mas, pela expressão (2.3), $1/k > f_0$, e, portanto, $M_2 \geq 1 + f_0 \left(1 - 1/f_0 \right) \geq f_0$.

Para facilitar a notação, define-se o conjunto $S_p = \{ i \in N : p/k \leq f_i < (p+1)/k \}$. Multiplicando (2.1) por M_1, e (2.7) por M_2, obtemos

$$\sum_{i \in N} a_i \left((1-\epsilon)/f_0 \right) x_i = a_0 \left((1-\epsilon)/f_0 \right), \text{ e}$$

$$\sum_{p=0}^{k-1} \sum_{i \in S_p} \left[(k+1) \lfloor a_i \rfloor + p + p/k - \left((1-\epsilon)/k f_0 \right) \left(k \lfloor a_i \rfloor + p \right) \right] x_i$$
$$\leq (k+1) \lfloor a_0 \rfloor - \left((1-\epsilon)/f_0 \right) \lfloor a_0 \rfloor \, .$$

Somando as duas expressões, temos

$$\sum_{p=0}^{k-1} \sum_{i \in S_p} \left[(k+1) \lfloor a_i \rfloor + p + \frac{(1-\epsilon)(f_i - p/k)}{f_0} + p/k \right] x_i$$
$$\leq (k+1) \lfloor a_0 \rfloor + (1-\epsilon).$$

A partir de (2.6), $f_i \geq p/k$, e, por definição, $f_0 < 1/k$. Então, $p \geq 1$ implica $f_i > f_0$. O conjunto S_0 é dividido em dois: $i \in S_0$ tal que $f_i \leq f_0$ e $i \in S_0$ tal que $f_i > f_0$. Assim, o lado esquerdo da expressão anterior pode ser reescrito como

$$\sum_{i \in S_0 : f_i \leq f_0} \left[(k+1)\lfloor a_i \rfloor + (1-\epsilon)\,\frac{f_i}{f_0} \right] x_i$$
$$+ \sum_{p=0}^{k-1} \sum_{i \in S_p : f_i > f_0} \left[(k+1)\lfloor a_i \rfloor + p + (1-\epsilon)\frac{f_i - p/k}{f_0} + \frac{p}{k} \right] x_i \tag{2.8}$$

O corte CGF é obtido aplicando o piso a todos os coeficientes da desigualdade acima. Na primeira somatória $f_i/f_0 \leq 1$ implica que $\lfloor (1-\epsilon) f_i/f_0 \rfloor = 0$. Assim o conjunto N_0 da desigualdade final do corte (2.4) está determinado.

Na segunda somatória observe que, como $f_i > f_0$,

$$\frac{f_i - p/k}{f_0} + \frac{p}{k} < 1 - \frac{p/k}{f_i} + \frac{p}{k}.$$

Como $p/k \geq 0$ e a partir da definição de S_p temos que $p/k < 1$,

$$\frac{f_i - p/k}{f_0} + \frac{p}{k} < 2.$$

Note ainda que a imposição

$$\frac{f_i - p/k}{f_0} + \frac{p}{k} > 1 \tag{2.9}$$

implica que

$$f_i > f_0 + \frac{p(1-f_0)}{k},$$

e portanto, o piso de (2.9) é 1. Este é o limitante inferior do conjunto N_p.

O conjunto N_{p+1} tem limitante inferior $f_i > f_0 + (p+1)(1-f_0)/k$. Portanto, o conjunto N_p é definido por todo f_i tal que $f_0 + p(1-f_0)/k < f_i \leq f_0 + (p+1)(1-f_0)/k$. Para determinar a expressão final do corte (2.4), toma-se o piso dos coeficientes de (2.8) e substitui-se p por $p-1$, de forma que as expressões anteriores podem ser reescritas, e a segunda somatória de (2.8) é limitada entre $p = 1$ e $p = k$.

Para demonstrar a relação de dominância utiliza-se a Definição 1.10. Dividindo a desigualdade (2.4) por $k+1$, o lado direito torna-se igual ao de (2.2), enquanto que, do lado esquerdo os coeficientes são maiores ou iguais aos de (2.2) e sempre existe um coeficiente estritamente maior para a mesma variável, pois $k \geq 1$. □

O corte do Teorema 2.1 pode ser derivado a partir de uma linha do tableau (1.1), gerando uma versão mais forte do corte fracionário de Gomory (1.3).

Teorema 2.2. Considere da i-ésima linha do tableau representada pela equação (1.1), em que $f_0 > 0$ e seja $k \geq 1$ o único inteiro tal que

$$1/(k+1) \leq f_0 < 1/k. \tag{2.10}$$

Seja *J* particionado em $k+1$ conjuntos da seguinte forma:

$$J_0 = \left\{ j \in J : f_j \leq f_0 \right\}, \text{ e, para } p = 1, \ldots, k,$$
$$J_p = \left\{ j \in J : f_0 + \frac{(p-1)(1-f_0)}{k} < f_j \leq f_0 + \frac{p(1-f_0)}{k} \right\}.$$

O corte Corte Fracionário Forte (CFF) é definido como a desigualdade

$$\sum_{j \in J_0} f_j x_j + \sum_{p=1}^{k} \sum_{j \in J_p} \left(f_j - p/(k+1) \right) x_j \geq f_0, \tag{2.11}$$

e domina o corte fracionário (1.3).

Demonstração: A partir da i-ésima linha do tableau,

$$x_i + \sum_{j \in J} \bar{a}_j x_j = \bar{a}_0, \tag{2.12}$$

o corte (2.4) é obtido. Note que a variável básica, x_i, tem coeficiente unitário, ou seja, $f_i = 0$. Portanto, $f_i \leq f_0$, e então, $i \in N_0$. As demais variáveis com coeficientes não nulos são não básicas. Portanto, $N_0 = \{i\} \cup \{J_0\}$ e $N_p = J_p$. Assim, a desigualdade do corte CGF (2.4) pode ser reescrita como

$$(k+1)x_i + \sum_{j \in J_0} (k+1) \left\lfloor \bar{a}_j \right\rfloor x_j + \sum_{p=1}^{k} \sum_{j \in J_p} \left((k+1) \left\lfloor \bar{a}_j \right\rfloor + p \right) x_j \leq (k+1) \left\lfloor \bar{a}_0 \right\rfloor. \tag{2.13}$$

A expressão do corte CFF é obtida dividindo-se a expressão acima por $k+1$ e subtraindo do resultado a equação do tableau (2.12).

Este corte domina o corte fracionário. O lado direito de (1.3) é igual ao de (2.11), e, como $k \geq 1$, então $p \geq 1$, e portanto existe um coeficiente para uma variável x_j que é estritamente menor em (2.11) que em (1.3). □

2.2.1 Testes Computacionais

O corte CFF foi testado por Letchford e Lodi (2002) utilizando a expressão (2.13). Esta forma foi escolhida porque gera menos erros de arredondamento que a sua forma final (2.11), pois possui coeficientes inteiros.

O procedimento de geração dos cortes proposto pelos autores pode ser dividido em dois passos. No primeiro passo, se a linha do tableau correspondente a uma variável básica fracionária tem lado

direito $< 1/2$, ela é multiplicada por -1, de forma que a parte fracionária do lado direito torne-se $> 1/2$. No segundo passo, o corte (2.13) é calculado com $k = 1$.

Cortes resultantes deste procedimento são comparados com os cortes fracionários de Gomory (1.2) e com os cortes fracionários pré-multiplicados por -1, como descrito anteriormente. Note que a pré-multiplicação por -1 não gera necessariamente um corte mais forte, ou seja, este é um procedimento heurístico.

Os testes são feitos com um conjunto de 50 instâncias do problema da mochila multidimensional gerados pelos próprios autores. O número de variáveis, n, é tal que $n \in \{5, 10, 15, 20, 25\}$, e o número de mochilas é $m \in \{5, 10\}$.

Os autores utilizam o CPLEX 7.0 para testar os planos de corte. São aplicados 1, 10 e 25 rounds de cortes ao nó raiz do método *branch-and-bound*. Os resultados mostram que os cortes CFF e os cortes de Gomory pré-multiplicados por -1 são mais eficientes no fechamento do gap de integralidade.

É reportado o número de nós necessários para encontrar a solução ótima quando esta não é obtida através da aplicação dos rounds ao nó raiz. Em geral, a aplicação dos cortes CFF resulta em um menor número de nós e o número de instâncias resolvidas no nó raiz também é maior.

Em um último experimento, é testado o algoritmo proposto originalmente por Gomory, em que é inserido apenas um plano de corte antes da reotimização. É selecionada a linha do tableau que tem o menor valor absoluto do lado direito para evitar erros numéricos. O uso dos cortes CFF possibilita um maior fechamento do gap de integralidade.

Entretanto, não há relação de dominância entre um corte CFF gerado a partir da pré-multiplicação da linha do tableau por -1 e o corte CFF gerado diretamente. Portanto, propõe-se a utilização do parâmetro *CFF-Kmax*. Este parâmetro determina o máximo valor de k considerado na geração de cortes CFF. Se o valor de k obtido pela aplicação das desigualdades (2.10) for maior que *CFF-Kmax*, então a linha do tableau é multiplicada por -1 antes da aplicação do corte fazendo com que $k = 1$.

Algoritmo 2: ALG-CFF

Dados: $x_i = \bar{a}_0 + \sum_{j \in J} \bar{a}_j(-x_j)$ e *CFF-Kmax*
Resultado: Corte CFF (2.13)

```
1 início
2     faça k = ⌈1/f_0⌉ − 1
3     se k > CFF-Kmax então
4         Multiplicar a linha do tableau por −1.
5         k = 1, por construção.
6     Calcular o corte CFF segundo expressão (2.13).
7 fim
```

Considere a linha do tableau cujo lado direito seja f_0. O parâmetro *CFF-Kmax* é limitado em função dos valores mínimo e máximo que deve ter f_0 para que o lado direito seja considerado fracionário. Se denominarmos este valor de *frac_rhs*, temos *frac_rhs* $\leq f_0 \leq 1 -$ *frac_rhs*, então, como $k = \lceil 1/f_0 \rceil - 1$,

$$1 \leq k \leq \lceil 1/\textit{frac\_rhs} \rceil - 1.$$

Por exemplo, se *frac_rhs* $= 0{,}1$, então $1 \leq k \leq 9$. Se *frac_rhs* $= 0{,}05$, $1 \leq k \leq 19$.

2.2.2 Exemplo

Exemplo 2.1. Considere o problema inteiro $X = P \cap Z^2_+$ em que $P = \{x \in R^2_+ : 6x_1 + 4x_2 \leq 9\}$. Para obter a desigualdade de Chvátal-Gomory (1.19), multiplica-se a inequação por $u = 4/7$ originando

$$3\frac{3}{7}x_1 + 2\frac{2}{7}x_2 \leq 5\frac{1}{7}$$

A desigualdade CG é dada pelo piso dos coeficientes da inequação anterior, $3x_1 + 2x_2 \leq 5$. Para o corte CGF (2.4),

$$k = \lceil 1/f_0 \rceil - 1 = 6.$$

Portanto $p = 1, ..., 6$, e $N_0 = \{i \in N : f_i \in \left[0, \frac{1}{7}\right]\}$, $N_1 = \{i \in N : f_i \in \left(\frac{1}{7}, \frac{2}{7}\right]\}$, $N_2 = \{i \in N : f_i \in \left(\frac{2}{7}, \frac{3}{7}\right]\}$, ..., $N_6 = \{i \in N : f_i \in \left(\frac{6}{7}, 1\right]\}$. Os conjuntos não vazios são, portanto, $N_1 = \{2\}$ e $N_2 = \{1\}$. A partir destes conjuntos calcula-se o corte CGF

$$23x_1 + 15x_2 \leq 35.$$

Considere agora, $u = 2/7$. A nova inequação é dada por

$$1\frac{5}{7}x_1 + 1\frac{1}{7}x_2 \leq 2\frac{4}{7}.$$

Note que a parte fracionária do lado direito desta expressão é $> 1/2$ e isto implica em $k = 1$. Assim, tem-se apenas os dois conjuntos $N_0 = \{i \in N : f_i \in \left[0, \frac{4}{7}\right]\}$ e $N_1 = \{i \in N : f_i \in \left(\frac{4}{7}, 1\right]\}$. Portanto, o corte CGF é $3x_1 + 2x_2 \leq 4$. A Figura 2.1 mostra as desigualdades no R^2. Note que, neste exemplo, a desigualdade CGF gerada para $k = 1$ domina as demais. □

2.3 K-cortes

Um k-corte corresponde à multiplicação de uma linha do tableau (1.1) por um parâmetro k positivo antes da geração do corte de Gomory GIM (1.13) para problemas de programação inteira mista ou dos cortes de Gomory GI (1.3) e GIF (1.14) para programação inteira. Garfinkel e Nemhauser (1972) apresentam k-cortes associados a GI em que k pode assumir valores inteiros e não inteiros, e discutem como determinar o melhor valor de k de modo a maximizar o lado direito do corte, e, portanto, torná-lo mais forte. Cornuéjols et al. (2003) analisam k-cortes associados a GIF, em que k é inteiro e apresentam resultados probabilísticos de dominância destes cortes em relação ao corte GIF. A proposição a seguir é devida a Cornuéjols et al. (2003).

Proposição 2.1. O k-corte, dado pelo corte GIF aplicado à linha do tableau (1.1) multiplicada por um k é válido para PI e dado por

$$\sum_{\tilde{f}_j \leq \tilde{f}_0} \frac{\tilde{f}_j}{\tilde{f}_0} x_j + \sum_{\tilde{f}_j > \tilde{f}_0} \frac{1 - \tilde{f}_j}{1 - \tilde{f}_0} x_j \geq 1, \tag{2.14}$$

em que $\tilde{f}_j = \tilde{a}_j - \lfloor \tilde{a}_j \rfloor, \forall\, j \in J$, $\tilde{f}_0 = \tilde{a}_0 - \lfloor \tilde{a}_0 \rfloor > 0$, $\tilde{a}_j = k\bar{a}_j, \forall\, j \in J$, e $\tilde{a}_0 = k\bar{a}_0$.

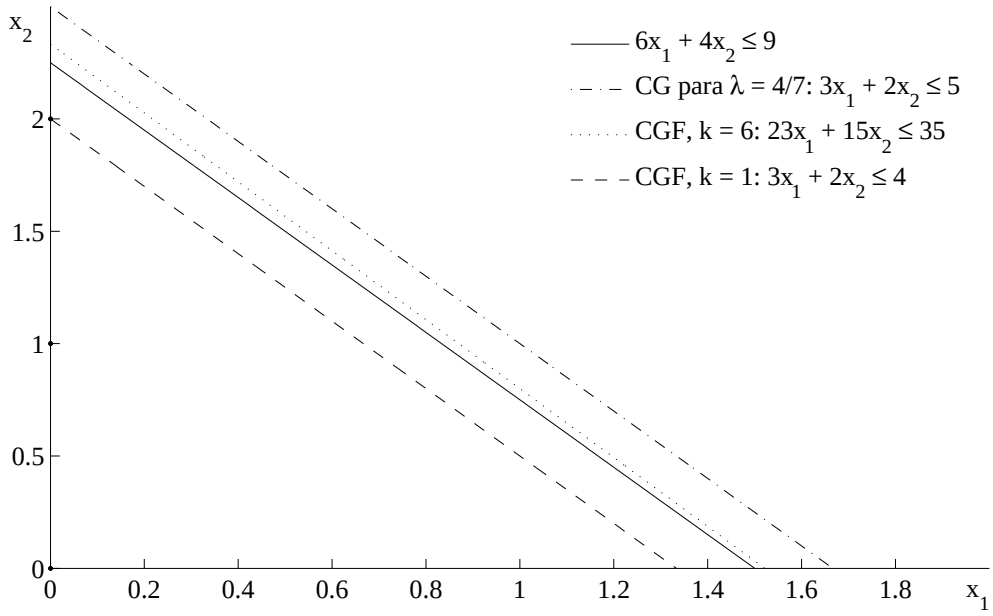

Figura 2.1: Comparação entre desigualdades CGF e CG.

Demonstração: Multiplicando-se ambos os lados da linha do tableau por $k \neq 0$, temos $kx_i = \tilde{a}_0 + \sum_{j \in J} \tilde{a}_j(-x_j)$. Substituindo-se kx_i por x'_i, temos $x'_i = \tilde{a}_0 + \sum_{j \in J} \tilde{a}_j(-x_j)$, que tem a mesma forma da linha do tableau original. Portanto, o processo de derivação do corte GIF pode ser aplicado, gerando o corte (2.14). □

Exemplo 2.2. Considere o seguinte problema binário de maximização,

$$\begin{array}{ll} \max & x_1 + x_2 \\ \text{s.a.} & 10x_1 + 2x_2 \leq 11, \\ & 2x_1 + 10x_2 \leq 11, \\ & x_i \in \{0,1\}, i = 1,2. \end{array}$$

Adicionando-se as variáveis de folga inteiras x_3, x_4, este problema pode ser reescrito como

$$\begin{array}{ll} \max & x_1 + x_2 \\ \text{s.a.} & 10x_1 + 2x_2 + x_3 = 11, \\ & 2x_1 + 10x_2 + x_4 = 11, \\ & x_i \in \{0,1\}, i = 1,2, \\ & x_i \in Z_+, i = 3,4. \end{array}$$

O tableau ótimo correspondente à relaxação linear deste problema é dado por

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.a.} \quad & x_1 + \frac{5}{48}x_3 - \frac{1}{48}x_4 = \frac{11}{12}, \\ & x_2 - \frac{1}{48}x_3 + \frac{5}{48}x_4 = \frac{11}{12}, \\ & x_i \in R_+, i = 1, 2, 3, 4. \end{aligned}$$

Os cortes GIF calculados a partir destas linhas, se colocados em função das variáveis originais, são dados por

$$\begin{aligned} & 72x_1 + 120x_2 \leq 132, \\ & 120x_1 + 72x_2 \leq 132. \end{aligned}$$

Os k-cortes para $k = 9$ são obtidos a partir da multiplicação das linhas do tableau por 9 seguida do cálculo do corte GIF. Aplicando este procedimento e novamente substituindo as variáveis de folga pelas variáveis originais, temos os dois cortes

$$\begin{aligned} & x_1 + 2x_2 \leq 2, \\ & 2x_1 + x_2 \leq 2. \end{aligned}$$

A Figura 2.2 mostra os cortes no R^2. O primeiro corte GIF intercepta a fronteira da região factível nos pontos $(1, 1/2)$ e $(1/6, 1)$, enquanto que o k-corte correspondente intercepta em $(1, 1/2)$ e $(0, 1)$. Portanto, o k-corte consegue restringir a região factível de forma mais eficiente. O mesmo pode ser observado para a segunda linha do tableau, pois o corte GIF intercepta a fronteira da região factível nos pontos $(1, 1/6)$ e $(1/2, 1)$, enquanto que a intersecção do k-corte com a região factível ocorre nos pontos $(1, 0)$ e $(1/2, 1)$. □

No exemplo anterior é ilustrada a relação entre os cortes GIF e os k-cortes. Para comparar estes dois cortes e também o corte fracionário GI, considere que qualquer um destes três cortes é descrito genericamente pela desigualdade $\gamma x \geq \gamma_0$, em que $\gamma_0 > 0$. Cornuéjols et al. (2003) definem a comparação entre cada coeficiente do corte como a seguir.

Definição 2.1. Sejam $\gamma x \geq \gamma_0$ e $\pi x \geq \pi_0$, dois cortes válidos para PI ou PIM, onde $\gamma, \pi \in R^n_+$, $\gamma_0 > 0$ e $\pi_0 > 0$. O corte $\gamma x \geq \gamma_0$ é *tão forte quanto* o corte $\pi x \geq \pi_0$, em relação à variável x_j, se $\gamma_j/\gamma_0 = \pi_j/\pi_0$. Este corte é *mais forte* em x_j se $\gamma_j/\gamma_0 \leq \pi_j/\pi_0$. O corte $\gamma x \geq \gamma_0$ é *estritamente mais forte* em x_j que o corte $\pi x \geq \pi_0$, se $\gamma_j/\gamma_0 < \pi_j/\pi_0$. □

A Figura 2.3 ilustra a relação γ_j/γ_0 em função de f_j para os três cortes mencionados, para $k = 5$ e $f_0 = 0,15$. Através desta figura é possível observar os intervalos em que há dominância de um corte em relação ao outro, para a variável x_j. Tais relações estão resumidas no Teorema 2.3.

Teorema 2.3. Seja a linha do tableau dada por

$$x_i = \bar{a}_0 + \sum_{j \in J} \bar{a}_j(-x_j), \tag{2.15}$$

em que $\bar{a}_0 = \lfloor \bar{a}_0 \rfloor + f_0$ e $\bar{a}_j = \lfloor \bar{a}_j \rfloor + f_j$, e $1 < f_0 < 1/2$. Então,

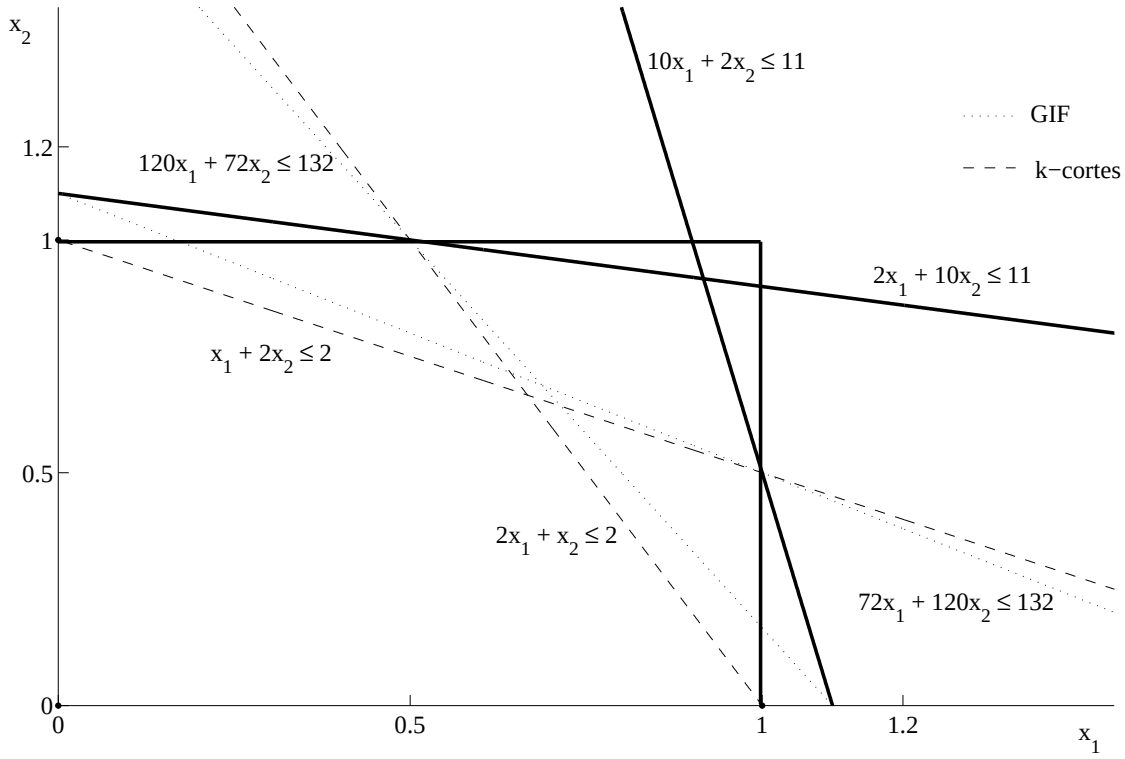

Figura 2.2: Comparação entre k-cortes e GIF no exemplo 2.2

(i) Quando $1 \leq k \leq \lfloor 1/f_0 \rfloor$, o k-corte gerado a partir desta linha é mais forte que o corte fracionário correspondente.

(ii) Admita que f_i é uniformemente distribuído no intervalo $[0, 1)$. Quando $2 \leq k \leq \lfloor 1/f_0 \rfloor$, a probabilidade de que o k-corte gerado a partir desta linha seja estritamente mais forte que o corte GIF correspondente em relação à variável x_j é $(1 - f_0)/2$, que é igual à probabilidade de que o corte GIF gerado a partir desta linha seja estritamente mais forte que o k-corte para a variável x_j. □

Note que no caso de (ii), o teorema implica que a probabilidade de que não se possa determinar qual dos dois cortes é mais forte é f_0. O teorema a seguir compara o conjunto de k-cortes $(2 \leq k \leq \lfloor \frac{1}{f_0} \rfloor)$ com o corte GIF.

Teorema 2.4. Admita que $\frac{1}{K+1} \leq f_0 < \frac{1}{K}$ para algum inteiro $K \geq 2$ e que f_i é uniformemente distribuído em $[0, 1)$. A probabilidade que algum k-corte, $(2 \leq k \leq K)$, gerado de alguma linha seja estritamente mais forte em x_j do que o GIF gerado a partir da mesma linha é $\frac{(K-1)(1-f_0)}{K}$, e a probabilidade de que o GIF gerado de alguma linha seja estritamente mais forte em x_j do que todos os k-cortes $(2 \leq k \leq K)$ gerados da mesma linha é $\frac{1-f_0}{K}$. □

Note que, pelo teorema anterior, para qualquer valor de $f_0 < 1/2$, a probabilidade de gerar um k-corte mais forte que um corte GIF é sempre maior que a probabilidade de gerar um corte GIF mais

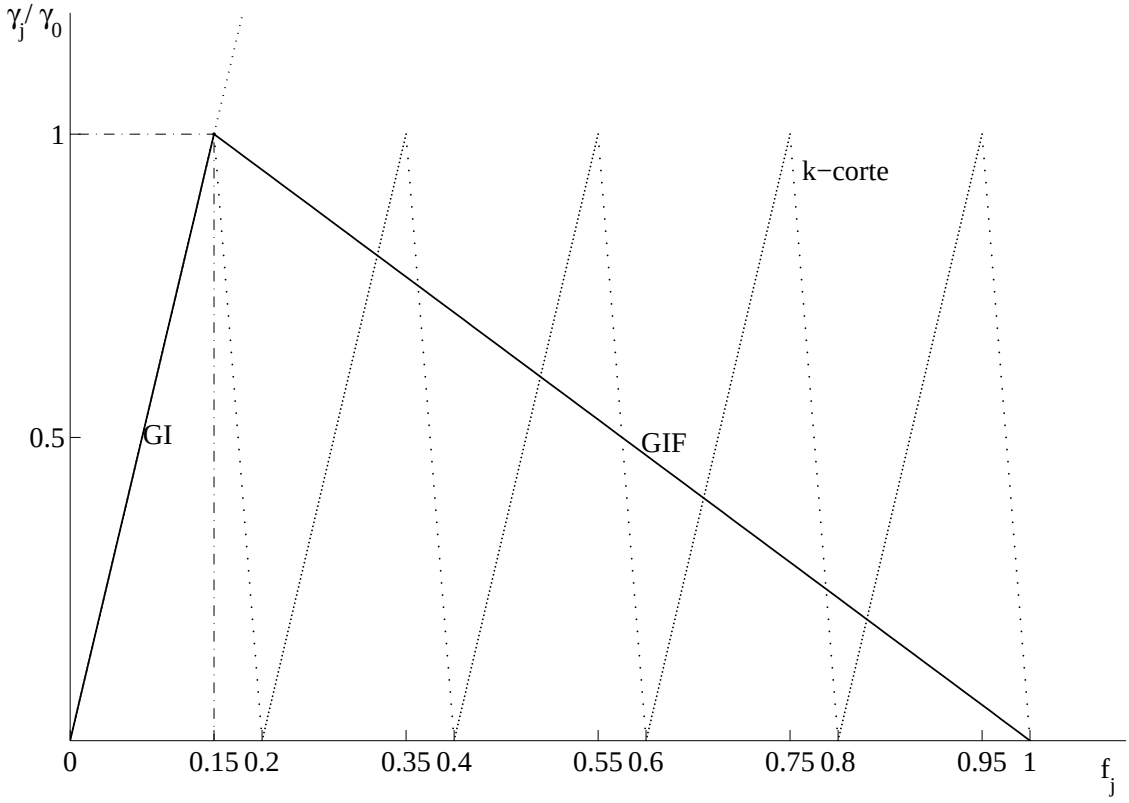

Figura 2.3: γ_j/γ_0 em função de f_j para GI, GIF e 5-corte

forte. Os teoremas 2.3 e 2.4 foram enunciados em (Cornuéjols et al., 2003) e estendidos para k-cortes em que $k > 1/f_0$.

2.3.1 Testes Computacionais

Em (Cornuéjols et al., 2003), testes computacionais são apresentados nos problemas da mochila com variáveis 0-1 e da mochila com variáveis inteiras limitada com uma restrição ou múltiplas restrições. Quatorze instâncias de cada problema são geradas com número de variáveis na faixa de 10 a 10.000. O critério de avaliação da qualidade dos cortes é a fração do gap de integralidade (diferença entre o valor do problema de programação inteira e o valor da relaxação linear) que é fechado ao se apertar a formulação com a inclusão de cortes.

No caso de uma restrição são aplicados o corte GIF e os k-cortes, tal que $1 \leq k \leq 5$, sem dominância por qualquer tipo de corte. A seguir, compara-se o GIF com a adição de todos os cortes com $1 \leq k \leq 10$ e todos os cortes com $1 \leq k \leq 50$. Os resultados mostram que a adição de vários k-cortes fecha uma fração muito maior do gap em relação a um único k corte.

No caso de múltiplas restrições foram geradas oito instâncias com 500 variáveis e com número de restrições igual a 5, 10, 50, e 100. Compara-se o GIF com a adição de todos os cortes com $1 \leq k \leq 10$

e todos os cortes com $1 \leq k \leq 50$. Os resultados são, na palavra dos autores, desapontadores: os gaps fechados diminuem com o aumento das restrições e melhoria da adição de grupos de k-corte é marginal em relação ao GIF. Os autores ressaltam que a utilidade de k-cortes deve ser verificada ao se resolver um problema até a solução ótima, por exemplo, em combinação com o método *branch-and-bound*.

Para testar os k-cortes neste trabalho utiliza-se o algoritmo ALG-K, a seguir. Neste algoritmo são utilizados valores de k tais que $k \leq \lfloor 1/f_0 \rfloor$. De acordo com o Teorema 2.3, para estes valores de k, o k-corte é mais forte que o corte fracionário (GI). Pelo Teorema 2.4 a probabilidade de que o k-corte seja mais forte que o corte GIF em x_j é maior do que o contrário, quando são levados em consideração todos os valores de $2 \leq k \leq K$. Portanto, o intervalo de valores possíveis para k é amostrado segundo o parâmetro MAXCL, que determina o máximo de k-cortes calculados para a mesma linha do tableau. Dentre os cortes calculados para a mesma linha com diferentes valores de k apenas um é selecionado para inserção no tableau.

Algoritmo 3: ALG-K

Dados: $x_i = \bar{a}_0 + \sum_{j \in J} \bar{a}_j(-x_j)$ e MAXCL
Resultado: k-corte, desigualdade (2.14)

```
início
    faça k_max = ⌊1/f_0⌋
    faça k = 1
    Δk = max {1, ⌊k_max/MAXCL⌋}
    enquanto k ≤ k_max faça
        Multiplicar a linha do tableau por k.
        Calcular o k-corte segundo (2.14).
        Armazenar em LK.
        k = k + Δk.
    Retornar o melhor corte em LK segundo critério de violação normalizada ou eficiência.
fim
```

2.4 Cortes Chvátal-Gomory-Nível

Os cortes CG-Nível (*CG-Tier*, na língua inglesa) são obtidos pela aplicação consecutiva do procedimento de derivação do corte Chvátal-Gomory para um problema de programação inteira. Suponha $\mathbf{x} \in \mathbb{Z}^n_+$, obedecendo à seguinte restrição associada à fonte do corte:

$$\sum_{i=1}^{n} a_i x_i = a_0. \tag{2.16}$$

Para obter o corte de Chvátal-Gomory, a equação fonte do corte é dividida por $\lambda \neq 0$ para obter

$$\sum_{i=1}^{n} (a_i/\lambda)\, x_i = a_0/\lambda.$$

Em seguida, toma-se o piso dos coeficientes do lado esquerdo originando

$$\sum_{i=1}^{n} \lfloor a_i/\lambda \rfloor \, x_i \leq a_0/\lambda.$$

Como tem-se apenas variáveis inteiras, podemos escrever:

$$\sum_{i=1}^{n} \lfloor a_i/\lambda \rfloor \, x_i \leq \lfloor a_0/\lambda \rfloor \, .$$

Como os coeficientes e variáveis da desigualdade anterior são inteiros, então a diferença entre os lados direito e esquerdo da equação também é inteira, e associada à variável de folga $z \in \mathbb{Z}_+$. Desta forma,

$$z + \sum_{i=1}^{n} \lfloor a_i/\lambda \rfloor \, x_i = \lfloor a_0/\lambda \rfloor \, . \tag{2.17}$$

Seja $d = -\lambda$, e, como $\lfloor -a \rfloor = - \lceil a \rceil$, a equação acima torna-se

$$z - \sum_{i=1}^{n} \lceil a_i/d \rceil \, x_i = - \lceil a_0/d \rceil \, . \tag{2.18}$$

Dividindo-se a equação (2.16) por $q \neq 0$, obtém-se a igualdade

$$\sum_{i=1}^{n} (a_i/q) \, x_i = a_0/q,$$

que somada à equação (2.18) resulta em

$$z - \sum_{i=1}^{n} \lceil a_i/d \rceil \, x_i + \sum_{i=1}^{n} (a_i/q) \, x_i = - \lceil a_0/d \rceil + a_0/q, \quad \text{ou}$$

$$z + \sum_{i=1}^{n} (a_i/q - \lceil a_i/d \rceil) \, x_i = a_0/q - \lceil a_0/d \rceil \, . \tag{2.19}$$

Os coeficientes q e d na equação (2.19) podem ser combinados de forma a dar origem a cortes clássicos de programação inteira. Com $q = \infty$ e $d < 0$, são obtidos os cortes associados ao método apenas com inteiros de Gomory e com $1/q = 1/d \in Z$ obtém-se os cortes fracionários associados ao método de formas inteiras de Gomory.

Para $q = 1$, obtém-se a equação do corte CG-Nível de nível 1 dada por

$$z + \sum_{i=1}^{n} (a_i - \lceil a_i/d \rceil) \, x_i = a_0 - \lceil a_0/d \rceil \, . \tag{2.20}$$

O corte CG-Nível consiste em aplicar o procedimento que parte da equação (2.16) e leva à equação (2.20) p vezes, dando origem a um corte de nível p. A cada nível, o parâmetro d é decrementado em uma unidade. A próxima proposição formaliza esta definição, e estabelece a forma geral a que este corte obedece.

Proposição 2.2. O corte CG-Nível de nível $p \in \mathbb{Z}_+$, é definido para $d \geq p \geq 1$ e tem a forma

$$z + \sum_{i=1}^{n} a_i^p x_i = a_0^p, \tag{2.21}$$

se observadas as seguintes definições recursivas para a_i^p:

$$a_i^0 = a_i, \tag{2.22}$$
$$d^k = d - k, \forall\ k \in \{0, 1, ..., p-1\} \text{ e} \tag{2.23}$$
$$a_i^k = a_i^{k-1} - \left\lceil \frac{a_i^{k-1}}{d^{k-1}} \right\rceil,\ \forall\ k \in \{1, ..., p\}\,. \tag{2.24}$$

Demonstração: As expressões (2.22) e (2.24) decorrem do desenvolvimento aplicado à equação (2.16) que leva à equação (2.20). Já a equação (2.23) determina o decremento de d em uma unidade a cada nível. Para o corte de nível 1, $d^0 = d$. Para o corte de nível p, $d^{p-1} = d - (p-1) = d - p + 1$. Por definição,

$$d \geq p \geq 1. \tag{2.25}$$

Desta forma, $d^{p-1} \geq 1$. Assim, temos que d^{k-1}, que aparece como divisor de a_i^p na equação (2.24), é o resultado de sucessivos decrementos do parâmetro d, de forma que $d^{k-1} \geq 1, \forall\ k \in \{1, ..., p\}$.

Demonstra-se, por indução, que o corte CG-Nível tem a forma indicada pela equação (2.21) para $1 \leq k \leq p$. Para $k = 1$, basta observar a equação (2.20). Suponha que a expressão seja válida para $k = n$. Então,

$$z + \sum_{i=1}^{n} a_i^n x_i = a_0^n.$$

Para obter o corte de nível $n+1$, o procedimento que leva da equação (2.16) à equação (2.20) deve ser reaplicado, tomando como equação original a do corte de nível n, e decrementando d. Isto é equivalente à aplicação das expressões (2.23) e (2.24).

Seja a'_z, o coeficiente de z no corte de nível n ($a_z = 1$) e z' a variável de folga que surge na derivação do corte Chvátal-Gomory. O corte do nível $n+1$ é

$$z' + a'_z z + \sum_{i=1}^{n+1} a_i^{n+1} x_i = a_0^{n+1},$$

tal que

$$a_i^{n+1} = a_i^n - \lceil a_i^n / d^n \rceil\,.$$

Em especial, para a variável z, o novo coeficiente será $a'_z = 1 - \lceil 1/d^n \rceil$. Além disso, $d^n = d - n$. Desta forma, a equação do novo corte torna-se

$$z' + \left(1 - \left\lceil \frac{1}{d-n} \right\rceil\right) z + \sum_{i=1}^{n+1} \left(a_i^n - \left\lceil \frac{a_i^n}{d-n} \right\rceil\right) x_i = a_0^n - \left\lceil \frac{a_0^n}{d-n} \right\rceil.$$

Por definição, temos $d \geq p \geq 1$. Como $p \geq n+1$, $d \geq n+1$. Portanto, $a_z^{'} = 0$, já que $d-n \geq 1$, e $\lceil 1/(d-n) \rceil = 1$. Ou seja, o coeficiente da variável de folga adicionada no nível n é zero no nível $n+1$, desde que $d \geq p$. Permanece, então, apenas a variável de folga $z^{'}$ correspondente ao corte de nível $n+1$. A nova equação do corte é

$$z^{'} + \sum_{i=1}^{n+1} \left(a_i^n - \left\lceil \frac{a_i^n}{d-n} \right\rceil \right) x_i = a_0^n - \left\lceil \frac{a_0^n}{d-n} \right\rceil .$$

Fazendo $z \equiv z^{'}$, o corte pode ser reescrito como

$$z + \sum_{i=1}^{n+1} \left(a_i^n - \lceil a_i^n / d^n \rceil \right) x_i = a_0^n - \lceil a_0^n / d^n \rceil \text{ , ou}$$

$$z + \sum_{i=1}^{n+1} a_i^{n+1} x_i = a_0^{n+1}. \quad \square$$

Para facilitar o desenvolvimento a seguir, o índice i é suprimido das equações (2.22) e (2.24). Considere ainda que $a^0 = a$, um dos coeficientes da equação fonte do corte. A proposição a seguir expressa o cálculo de a^p.

Proposição 2.3. Seja $r = \lceil d - \beta \rceil$, em que $\beta = \lceil a/d \rceil \, d - a$. Então,

$$a^p = \begin{cases} a - p \lceil a/d \rceil , & \text{se } 1 \leq p \leq r, \\ a - p \lceil a/d \rceil + (p-r) , & \text{se } p \geq r+1. \end{cases}$$

Demonstração: A demonstração desta proposição está na seção A.1 do Apêndice A. $\square$

A proposição a seguir mostra o comportamento dos coeficientes de corte em função de $d \geq p$.

Proposição 2.4. Seja $p \in \mathbb{Z}_+$ fixo e tal que $d \geq p \geq 1$. Então, para qualquer coeficiente a, o coeficiente correspondente, a^p, em (2.21), é função de d e varia como mostrado a seguir.

(i) Se $a > 0$, então

$$a - \lceil a \rceil \leq a^p \leq \begin{cases} a - p, & \text{se } p \leq \lceil a \rceil , \\ a - \lceil a \rceil , & \text{se } p \geq \lceil a \rceil + 1, \end{cases} \tag{2.26}$$

em que a^p é uma função não-decrescente de d e o limitante superior de a^p é atingido para qualquer $d \geq a$.

(ii) Se $a \leq 0$, então

$$a \leq a^p \leq a - \lceil a \rceil , \tag{2.27}$$

em que a^p é uma função não-crescente de d e o limitante inferior de a^p é atingido para qualquer $d > p - a - 1$.

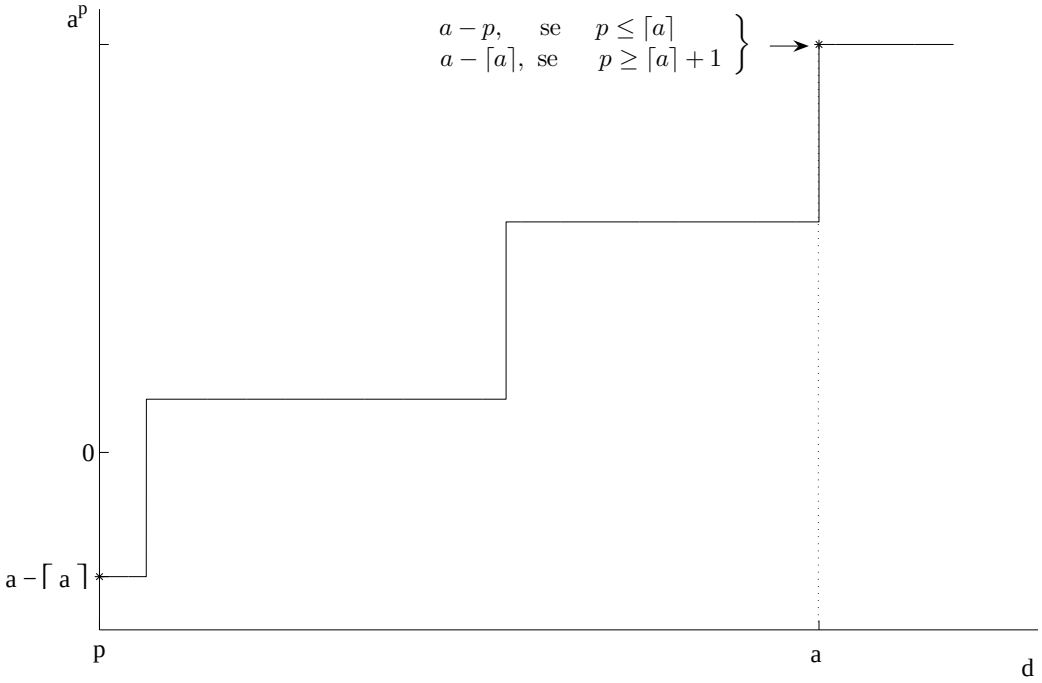

Figura 2.4: a^p em função de d para $a > 0$

Demonstração: A demonstração desta proposição está na na seção A.2 do Apêndice A. □

As figuras 2.4 e 2.5 ilustram o comportamento determinado pela proposição em questão para coeficientes positivos e negativos.

Note que a^p, na Proposição 2.3, assume valores discretos. Basta notar que qualquer variação no parâmetro d pode acarretar variação no termo $\lceil a/d \rceil$ e no termo r, ambos inteiros. Portanto, a variação final é sempre inteira. Isto explica o gráfico em forma de escada nas figuras 2.4 e 2.5.

Os resultados já obtidos permitem vislumbrar um método para síntese de cortes do tipo CG-Nível. Pode-se determinar o valor almejado para o lado direito da equação do corte, por exemplo, $a_0^p = y$. O parâmetro d deve, então, ser escolhido de forma a maximizar os coeficientes do lado esquerdo da equação (2.21). Se $a_i > 0$ para todo i, na equação original, então, para qualquer p, a Proposição 2.4 deixa claro que o corte mais forte será obtido para o maior valor possível de $d \geq p$, tal que $a_0^p = y$. Analogamente, se $a_i \leq 0$ para todo i, na equação original, então, o menor valor de $d \geq p$ origina o corte mais forte. A proposição a seguir determina os valores de $d \geq p$, de forma que $a_0^p = y$. Assim é possível escolher o melhor valor de d para os casos citados, bem como tentar direcionar a escolha de d em casos em que os coeficientes a_i assumem valores negativos e positivos ao mesmo tempo.

Proposição 2.5. Seja $p \geq 1 \in \mathbb{Z}$ e y tal que $a - y \in \mathbb{Z}$. Sejam $\alpha = \lceil (a-y)/p \rceil - 1$ e $\delta = p(\alpha + 1) - (a - y)$. Então, $a^p = y$ se e somente se $d \geq p$, satisfazendo as seguintes condições:

Para $a > 0$:

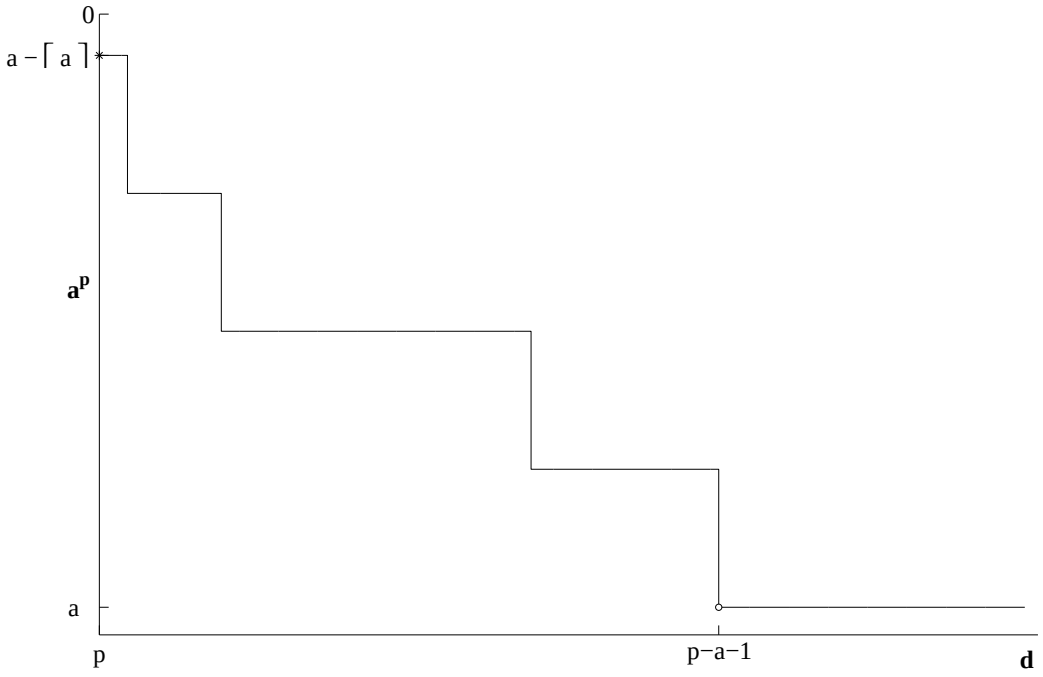

Figura 2.5: a^p em função de d para $a \leq 0$

(i) $d \geq a/\left(\alpha + 1\right)$,

(ii) $d < p + \left(y + 1\right)/\alpha$, se $\alpha \neq 0$, e

(iii) $d \geq p + y/\alpha$, se $\alpha \neq 0$ e $\delta > 0$.

Para $a \leq 0$:

(i) $d > p + (y + 1)/\alpha$, e

(ii) $d \leq p + y/\alpha$, se $\delta > 0$.

Demonstração: A demonstração desta proposição está na seção A.3 do Apêndice A, onde mostra-se também que a condição (iii) é sempre mais restritiva que (i), para $a > 0$.

Substituindo α na expressão de δ e dividindo a expressão resultante por p, obtém-se

$$\delta/p = \lceil (a - y)/p \rceil - (a - y)/p \geq 0.$$

Portanto, δ é p vezes a diferença entre o teto de um número e ele mesmo, e então, $\delta \geq 0$. Observe que $p > \delta$, pois $\delta/p < 1$. Portanto, conclui-se que $p > \delta \geq 0$. Pela Proposição 2.3, $a^p = y$ implica $a - y \in \mathbb{Z}$, e portanto, δ também é inteiro, de forma que $p - 1 \geq \delta \geq 0$. □

A próxima proposição mostra a variação no coeficiente do corte, a^p, quando d assume um valor arbitrariamente próximo de um limitante superior ou inferior que não pode ser atingido. Ela será

aplicada no o cálculo do coeficiente a_i^p quando d estiver restrito ao limitante (ii), para $a_i > 0$, ou (i) para $a_i \leq 0$.

Proposição 2.6. Dado d^*, se $d = d^* - \epsilon$, para $\epsilon > 0$ e suficientemente pequeno, então

(i) $\lceil a/d \rceil = \lfloor a/d^* \rfloor + 1$ e $r = \lfloor a - d^* \lfloor a/d^* \rfloor \rfloor + 1$, para $a > 0$;

(ii) $\lceil a/d \rceil = \lceil a/d^* \rceil$ e $r = \lceil a - d^* (\lceil a/d^* \rceil - 1) \rceil$, para $a \leq 0$.

Se $d = d^* + \epsilon$, então a condição (i) é válida para $a \leq 0$ e a condição (ii) é válida para $a > 0$.

Demonstração: Os resultados desta proposição decorrem de que, para $\epsilon > 0$ suficientemente pequeno, tem-se $\lceil z - \epsilon \rceil = \lceil z \rceil$ e $\lceil z + \epsilon \rceil = \lfloor z \rfloor + 1$, para um número z qualquer. Suponha, por exemplo, $d = d^* - \epsilon$ e $z = a/d$. Desta forma, $\lceil a/d \rceil = \lceil a/(d^* - \epsilon) \rceil$. Para $a > 0$, $\lceil a/(d^* - \epsilon) \rceil = \lceil a/d^* + \epsilon' \rceil$ = $\lceil a/d^* \rceil + 1$. Para $a \leq 0$, $\lceil a/(d^* - \epsilon) \rceil = \lceil a/d^* - \epsilon' \rceil$ = $\lceil a/d^* \rceil$. O mesmo raciocínio é válido para os demais casos. □

Na próxima seção é analisada a geração de cortes CG-Nível, que são determinados por um par de parâmetros p e d. A forma geral à qual obedecem os coeficientes do corte e sua variação máxima estão na Proposição 2.3 e na Proposição 2.4, respectivamente. Esta última proposição mostra a variação do coeficiente a^p em relação ao parâmetro d. Entretanto, na geração de cortes pode ser necessário comparar dois conjuntos de parâmetros em que ambos variam ao mesmo tempo. A proposição a seguir fornece argumentos para comparação de cortes quando ocorre variação simultânea em p e d.

Proposição 2.7. Se p e d são incrementados de forma que se mantenha $d \geq p$ e o incremento de p é no mínimo igual ao incremento de d, então

(i) O termo a^p diminui ou permanece o mesmo para $a > 0$;

(ii) O termo a^p aumenta ou permanece o mesmo para $a \leq 0$.

Demonstração: A demonstração desta proposição está na seção A.4 do Apêndice A. □

2.4.1 Variação de Parâmetros

A partir da Proposição 2.5 é possível desenvolver uma estratégia de escolha de parâmetros p e d de forma a gerar cortes CG-Nível mais fortes.

O passo inicial para geração de um corte CG-Nível é a determinação do valor y que o lado direito deste corte deve assumir. Tal definição é detalhada mais adiante e está ligada à determinação do parâmetro p. Supõe-se, por hora, que o valor de p e do lado direito do corte são dados. Então, com o lado direito do corte, a_0^p, fixado em um determinado valor y, o corte será mais forte quanto maiores forem os coeficientes positivos e negativos do lado esquerdo deste corte.

Parâmetro d

Mesmo dado o valor de y e p, não é possível de determinar para o corte CG-Nível um parâmetro ótimo d, para uma linha qualquer do tableau, a não ser que os coeficientes sejam de um único sinal. Para um dado valor de p e y, a Proposição 2.4 mostra que, se $a_i \geq 0$, para todo i, o corte mais forte é determinado pelo maior valor de $d \geq p$ tal que $a^p = y$. De modo análogo, se $a_i \leq 0$, para todo i, o corte mais forte é dado pelo menor valor de $d \geq p$ tal que $a^p = y$.

Isto acontece porque coeficientes positivos se comportam de forma oposta a coeficientes negativos em relação à variação de d, como mostra a Proposição 2.4, ilustrada nas figuras 2.4 e 2.5. O aumento de d implica na diminuição do coeficiente a^p para os coeficientes negativos (tornam-se mais negativos ou permanecem iguais) e no seu aumento para os coeficientes positivos (tornam-se mais positivos ou permanecem iguais).

Esta é a principal motivação para a definição de três diferentes estratégias de escolha do parâmetro d. Em todas elas é levado em conta o fato de o lado direito da linha original, a_0, ser positivo nas linhas do tableau.

Maximizar coeficientes positivos. Como já foi observado, o coeficiente a_i^p que aparece na equação do corte é uma função não-decrescente de d, se $a_i > 0$. Portanto, é desejado o maior valor de d possível. Desta forma, é desenvolvido um algoritmo para encontrar este valor máximo, d_{max}.

Denomina-se d_i a variável utilizada para expressar as restrições sobre o parâmetro d impostas pelo i-ésimo coeficiente da linha do tableau. Para um determinado par (p, y), e $a_0 > 0$, a Proposição 2.5 determina os intervalos válidos para d_0. Se $\alpha = 0$, existe apenas um limitante inferior $d_0 > a_0$, e todos os coeficientes positivos do lado esquerdo são considerados. Pela Proposição 2.4, cada coeficiente atinge seu valor máximo para $d_i \geq a_i$. Portanto, $d_i = a_i$ e $d_{max} = \max\{d_i\}, i = 0, ..., n$.

Observe que poderia ser usado qualquer valor de d_i tal que $d_i \geq a_i$. Como deseja-se maximizar os coeficientes positivos afetando minimamente os coeficientes negativos, d_i é o mínimo valor tal que $d_i \geq a_i$, ou seja, $d_i = a_i$.

Se $\alpha \neq 0$, existe um limitante superior $d_0 < p + (y + 1)/\alpha$. Assim, tomamos um valor de d mais próximo deste limitante: $d_{max} = p + (y+1)/\alpha - \epsilon$. A partir da aplicação da Proposição 2.6 é possível calcular os coeficientes da desigualdade do corte. O algoritmo *ALGDmax* resume estes passos.

Maximizar coeficientes negativos. O lado direito da equação original é sempre positivo, pois supõe-se que esta seja uma linha do tableau. Portanto, da mesma forma que foi feito para os coeficientes positivos, para um determinado par (p, y), e $a_0 > 0$, a Proposição 2.5 determina os intervalos válidos para d_0. Portanto, $d_0 = p + y/\alpha$, se $\alpha \neq 0$ e $\delta > 0$ e $d_0 = a_0/(\alpha + 1)$, caso contrário.

Este é o valor mínimo para o parâmetro d, $d_{min} = d_0$. Como este limitante deve ser respeitado não é necessário considerar os demais coeficientes, pois não é possível encontrar um valor menor que d_{min} que gere $a_0^p = y$. O algoritmo *ALGDmin* resume estes passos.

Estratégias de duas fases. As estratégias anteriores levam em consideração coeficientes de um único sinal. Uma estratégia alternativa é dividir o cálculo do parâmetro d em duas fases. Na primeira fase, a escolha do parâmetro d é feita de modo a maximizar coeficientes de um único sinal. Se a linha do tableau contém um número maior de coeficientes negativos, aplica-se o algoritmo ALGDmin, caso contrário, aplica-se o algoritmo ALGDmax, calculando d_{min} ou d_{max}, respectivamente.

Algoritmo 4: ALGDMax

Dados: o par (p, y), e $a_0 > 0$
Resultado: d_{max}

início
  faça $d_{max} = -\infty$
  faça $a_0^p = y$ na Proposição 2.5 .Então,
  $\alpha = \lceil (a_0 - y)/p \rceil - 1$
  se $\alpha = 0$ **então**
    para todo $i = 0..n$ **faça**
      Pela Proposição 2.4, temos o máximo de a_i^p para $d_i \geq a_i$
      faça $d_i = a_i$
      se $d_i > d_{max}$ **então**
        faça $d_{max} = d_i$
  senão
    $d_0 < p + (y + 1)/\alpha$
    faça $d_{max} = p + (y + 1)/\alpha - \epsilon$
fim

Algoritmo 5: ALGDMin

Dados: o par (p, y), e $a_0 > 0$
Resultado: d_{min}

início
  faça $a_0^p = y$ na Proposição 2.5. Então,
  $\alpha = \lceil (a_0 - y)/p - 1 \rceil$
  $\delta = p(\alpha + 1) - (a_0 - y)$
  se $\alpha \neq 0$ *e* $\delta \geq 0$ **então**
    $d_0 = p + y/\alpha$
  senão
    $d_0 = a_0/(\alpha + 1)$
  faça $d_{min} = d_0$
fim

Uma vez determinado o valor de d, e como $a_0^p = y$ e p são conhecidos, basta aplicar a Proposição 2.3 e obter a_i^p, para $i = 1, ..., n$. Como o ponto de partida foi o coeficiente do lado direito, este já está previamente determinado: $a_0^p = y$.

Na segunda fase é estudada a possibilidade de modificação do valor de d. O objetivo é aumentar o valor dos coeficientes negativos sem modificar os positivos ou aumentar o valor dos coeficientes positivos sem modificar os negativos.

Inicialmente é analisada a possibilidade de aumento dos coeficientes negativos. O procedimento está representado no algoritmo *ALGDmaxmin*. A Proposição 2.5 deve ser aplicada sobre todos os coeficientes *positivos* da linha original $(a_i > 0)$, determinando os intervalos a que deve obedecer o parâmetro d_i, de forma a manter a_i^p inalterados. Este procedimento retorna D_{inf}, que é o menor d que mantém os coeficientes positivos inalterados e pode gerar os maiores valores de a_i^p para os coeficientes negativos. Se este valor for igual a d_{max}, obtido por ALGDmax na primeira fase, então não houve melhoria no valor de d na segunda fase. Observe que se $\alpha = 0$ no algoritmo ALGDmax, então d_{max} já é o mínimo d que maximiza todos os coeficientes positivos e torna-se desnecessária a aplicação do algoritmo ALGDmaxmin.

Os coeficientes negativos são calculados a partir da Proposição 2.3, utilizando $d = D_{inf}$. Como o algoritmo foi projetado para manter inalterados os coeficientes positivos, estes não precisam ser recalculados. O novo corte obtido a partir de D_{inf} é no mínimo tão forte quanto aquele obtido pelo parâmetro d_{max} obtido a partir de ALGDmax na primeira fase.

Algoritmo 6: ALGDmaxmin (estratégia de duas fases para $a_i > 0$)

Dados: o par (p, d_{max}), obtido a partir de ALGDmax
Resultado: D_{inf}

```
início
    faça D_inf = −∞
    para todo i = 0..n faça
        se a_i > 0 então
            calcule a_i^p, para d = d_max, aplicando a Proposição 2.3
            faça y = a_i^p na Proposição 2.5
            α = ⌈(a_i − a_i^p)/p − 1⌉
            δ = p(α + 1) − (a_i − a_i^p)
            se α ≠ 0 e δ ≠ 0 então
                d_inf = p + a_i^p/α
            senão
                d_inf = a_i/(α + 1)
            Atualizar o intervalo válido:
            se d_inf > D_inf então
                D_inf = d_inf
fim
```

O mesmo pode ser feito considerando os coeficientes negativos fixos e prosseguindo com o ajuste dos coeficientes positivos (maximização). A Proposição 2.5 deve ser aplicada sobre todos os coefi-

cientes *negativos* da linha original, determinando os intervalos a que deve obedecer o parâmetro d_i, de forma a manter a_i^p inalterados.

No Algoritmo *ALGDminmax*, a seguir, d_{sup} é determinado para cada coeficiente negativo $(a_i < 0)$. O mínimo deste valor para todo i é acumulado em D_{sup} e representa o valor de d que maximiza os coeficientes positivos mantendo os negativos inalterados. Se este valor for igual a d_{min}, retornado por ALGDmin, então não houve melhoria no valor de d.

Os coeficientes positivos do corte podem ser calculados, sem alteração dos negativos. O novo corte é no mínimo tão forte quanto o corte gerado por ALGDmin.

Algoritmo 7: ALGDminmax (estratégia de duas fases para $a_i < 0$)

Dados: o par (p, d_{min}), obtido a partir de ALGDmin
Resultado: D_{sup}

início
  Faça $D_{sup} = +\infty$
  para todo $i = 0..n$ **faça**
    se $a_i < 0$ **então**
      Calcule a_i^p, para $d = d_{min}$ aplicando a Proposição 2.3
      Faça $y = a_i^p$ na Proposição 2.5
      $\alpha = \lceil (a_i - a_i^p)/p - 1 \rceil$
      $\delta = p(\alpha + 1) - (a_i - a_i^p)$
      se $\delta > 0$ **então**
        $d_{sup} = p + a_i^p/\alpha$
      senão
        $d_{sup} = +\infty$
      Atualizar o intervalo válido:
      se $d_{sup} < D_{sup}$ **então**
        $D_{sup} = d_{sup}$
fim

Parâmetros p e y

Até aqui foi discutida a obtenção do parâmetro d, supondo p e y previamente determinados. É necessário esclarecer a relação existente entre estes dois parâmetros para determinar uma estratégia de escolha.

Os parâmetros p e y (nível e lado direito do corte) estão diretamente ligados pela Proposição 2.4. Nesta proposição fica clara a relação entre a variação máxima de p e o coeficiente do corte a^p. Se $p \geq \lceil a_0 \rceil + 1$, então $y = a_0^p = a_0 - \lceil a_0 \rceil$. Portanto y está fixo em em seu valor mínimo. Para evitar esta situação devemos ter $p \leq \lceil a_0 \rceil$, e, portanto, $1 \leq p \leq \lceil a_0 \rceil$.

Assim temos $y \leq a_0 - p$. Portanto, dado o nível do corte, p, y estará limitado por

$$a_0 - \lceil a_0 \rceil \leq y \leq a_0 - p. \tag{2.28}$$

Pela Proposição 2.5, $a_0 - y \in Z$. Pelas inequações (2.28), $a_0 - y \leq \lceil a_0 \rceil$ e $a_0 - y \geq p$. Assim, dado o nível do corte p_t,

$$a_0 - y = \{p_t, p_t + 1, \ldots, \lceil a_0 \rceil\}. \tag{2.29}$$

O parâmetro p_t, determina, portanto, o número de valores que podem ser designados a y, dado a_0. Quanto maior p_t, menor o número de valores que y pode assumir. Mas $1 \leq p_t \leq \lceil a_0 \rceil$, e, portanto, para definir uma busca por cortes mais fortes dentro deste intervalo de valores válidos de p_t, define-se p_{min} e p_{max}. O parâmetro p_{max} é o máximo valor assumido por p_t, e portanto, $p_{max} = \min \{p_{max}, \lceil a_0 \rceil\}$. O parâmetro p_{min} é o mínimo valor de p_t utilizado na geração de cortes, e portanto, p_{min} = max $\{1, p_{min}\}$. O valor p_{max} procura limitar a geração de cortes para valores muito grandes de $\lceil a_0 \rceil$. Já p_{min} pode ser usado para obrigar a geração de cortes de maior nível.

Para cada p_t contido no intervalo $[p_{min}, p_{max}]$, a equação (2.29) pode determinar muitos valores possíveis para y, e, se for gerado um corte para cada um destes valores, o tempo total gasto na geração de cortes pode ser proibitivo. Assim, define-se o parâmetro MAXPY, que é o máximo número de cortes gerados por valor do parâmetro p_t. O intervalo de valores possíveis é amostrado segundo o incremento Δp, definido como

$$\Delta p = \max \left\{ 1, \left\lfloor \frac{p_{max} - p_{min} + 1}{MAXPY} \right\rfloor \right\}.$$

O algoritmo *ALGpy*, a seguir, formaliza os passos descritos acima. A saída deste algoritmo é um conjunto de valores de p e y, acumulados em vec_{py}.

Algoritmo 8: ALGpy

Dados: $[p_{min}, p_{max}]$, e $a_0 > 0$
Resultado: vec_{py}

```
início
    faça p_max = min {p_max, ⌈a_0⌉}
    faça p_t = max {1, p_min}
    faça Δp = max{1, ⌊( p_max − p_min + 1 )/MAXPY⌋}
    enquanto p_t ≤ p_max faça
        faça p = p_t
        enquanto p ≤ p_max faça
            calcule o valor do lado direito:
            faça y = a_0 − p
            armazene o par (p_t, y) em vec_py
            p = p + Δp
        atualize o parâmetro p_t:
        p_t = p_t + 1
fim
```

2.4.2 Geração de Cortes Chvátal-Gomory-Nível

A geração de um corte CG-Nível dá-se através da escolha de um par de parâmetros (p, d). Dada uma linha do tableau, os parâmetros p e y são obtidos a partir do algoritmo ALGpy, enquanto o valor de d é obtido através de um algoritmo de uma ou duas fases. Na primeira fase é utilizado o algoritmo ALGDmin ou ALGDmax. Se houver segunda fase, são utilizados, respectivamente, ALGDminmax e ALGDmaxmin.

O algoritmo *ALG-CGN* (Chvátal-Gomory-Nível) descreve a geração do corte para uma determinada linha do tableau. A lista de parâmetros (p, y) gerada por ALGpy é armazenada em vec_{py}. Cada par de parâmetros (p_i, y_i) é considerado para a determinação do parâmetro d. Se a maioria dos coeficientes a_i for maior que zero, então aplica-se na primeira fase ALGDmax e o resultado (p_i, y_i, d_{max}) é armazenado em uma lista de parâmetros, LPD. Se a maioria dos coeficientes a_i for menor que zero, então aplica-se na primeira fase ALGDmin e o resultado (p_i, y_i, d_{min}) é armazenado em LPD.

Em seguida, a Proposição 2.7 é utilizada para descartar parte da lista LPD. A motivação principal é evitar perda de tempo de processamento no cálculo dos coeficientes de cortes para parâmetros que indiquem um corte mais fraco. Observe ainda que, para os algoritmos de determinação de parâmetros de duas fases, ALGDminmax e ALGDmaxmin, há cálculo de coeficientes de corte na segunda fase. Esta é a motivação para que o descarte seja feito antes de sua aplicação.

Para efetuar o descarte de parâmetros é necessário analisar a lista LPD. Este procedimento também exige tempo de processamento. Entretanto, o tempo economizado no cálculo de cortes supera o tempo gasto no descarte de parâmetros.

A Proposição 2.7 indica a evolução oposta dos coeficientes negativos e positivos, com a variação dos parâmetros p e d. Suponha dois conjuntos de parâmetros, encontrados nas iterações t_1 e t_2 do algoritmo ALGpy, para o mesmo valor de y: (p_{t_1}, d_{t_1}) e (p_{t_2}, d_{t_2}), em que $p_{t_1} \leq p_{t_2}$. De acordo com a Proposição 2.4, se $p_{t_2} - p_{t_1} \geq d_{t_2} - d_{t_1}$, então, para os coeficientes positivos, $a_i^{p_{t_1}} \geq a_i^{p_{t_2}}$. Portanto, o corte gerado a partir do segundo conjunto de parâmetros nunca é mais forte, podendo ser mais fraco que aquele gerado pelo primeiro conjunto, em relação aos coeficientes positivos. Entretanto, para os coeficientes negativos, a análise é contrária. E assim, apenas o balanço final dos coeficientes do corte poderia indicar qual deles é mais forte.

Fica claro, portanto, que não há relação de dominância entre o corte que seria gerado a partir dos parâmetros descartados e os demais. Portanto, a motivação principal para o descarte de parâmetros é diminuir o tempo de processamento, evitando cálculo de coeficientes para cortes que tendem a ser mais fracos segundo a estratégia escolhida.

A comparação entre os cortes calculados segundo cada conjunto de parâmetros não descartado é feita segundo um de dois possíveis critérios: violação normalizada expressa na Definição 1.20 ou eficiência, Definição 1.22.

O algoritmo ALG-CGN, a seguir, resume todo o procedimento de geração de cortes para uma linha do tableau.

2.4.3 Exemplos

Exemplo 2.3. Considere o exemplo proposto por Ben-Israel e Charnes (1962) com o objetivo de demonstrar a limitação dos cortes de Gomory. Seja $(x_1, x_2) \in Z_+^2$ tal que $3x_1 + 7x_2 \leq 26$. Adicionando

Algoritmo 9: ALG-CGN

Dados: $\sum_{i=1}^{n} a_i x_i = a_0$
Resultado: $\sum_{i=1}^{n} a_i^p x_i \leq a_0^p$

início
  vec$_{py}$ = ALGpy $([p_{min}, p_{max}], a_0)$
  para todo $(p_i, y_i) \in$ *vec$_{py}$* **faça**
    se *Maioria dos coeficientes* $a_i > 0$ **então**
      Aplicar $d_{max} =$ AlGDmax (p_i, y_i, a_0)
      Armazenar (p_i, y_i, d_{max}) em LPD
    senão
      Aplicar $d_{min} =$ AlGDmin (p_i, y_i, a_0)
      Armazenar (p_i, y_i, d_{min}) em LPD
  Aplicar a Proposição 2.7 para descartar parâmetros:
  se *Maioria dos coeficientes* $a_i > 0$ **então**
    Descartar todo $(p_2, y, d_2) \in$ LPD se existe um outro conjunto (p_1, y, d_1) tal que $p_2 - p_1 \geq d_2 - d_1$.
  senão
    Descartar todo $(p_1, y, d_1) \in$ LPD se existe um outro conjunto (p_2, y, d_2) tal que $p_2 - p_1 \geq d_2 - d_1$.
  Aplicar segunda fase:
  se *Maioria dos coeficientes* $a_i > 0$ **então**
    para todo $(p_i, d_{max}) \in$ *LPD* **faça**
      $d_{max} =$ AlGDmaxmin (p_i, d_{max})
  senão
    para todo $(p_i, d_{max}) \in$ *LPD* **faça**
      $d_{min} =$ AlGDminmax (p_i, d_{min})
  Para todos os parâmetros em LPD determinar $\sum_{i=1}^{n} a_i^p x_i \leq a_0^p$ e escolher o melhor corte segundo violação normalizada ou eficiência.
fim

a variável de folga inteira, s, temos

$$s + 3x_1 + 7x_2 = 26.$$

Uma faceta da região factível da relaxação linear deste problema é dada por

$$z + x_1 + 3x_2 = 10, \tag{2.30}$$

em que $z \in Z_+$ é variável de folga adicionada à desigualdade.

Note que não é possível gerar esta faceta através dos cortes de Gomory, pois não existe valor de λ tal que

$$z + \lfloor 1/\lambda \rfloor s + \lfloor 3/\lambda \rfloor x_1 + \lfloor 7/\lambda \rfloor x_2 = \lfloor 26/\lambda \rfloor$$

gere a faceta mencionada. Aplicando os cortes CG-Nível, entretanto, é possível gerar esta faceta. Para tanto, obtém-se o corte em que o coeficiente de x_1 e x_2 é positivo, e que tenha $a_0^p = 10$. Para todos os coeficientes, a Proposição 2.4 mostra o máximo e mínimo valores assumidos por a_i^p. Note que os coeficientes da equação original estão limitados por

$$0 \leq a_1^p \leq 3 - p, p \leq 3,$$
$$0 \leq a_2^p \leq 7 - p, p \leq 7,$$
$$0 \leq a_0^p \leq 26 - p, p \leq 26.$$

Portanto, qualquer valor de $p \geq 3$ implica em $a_1^p = 0$, e então, os possíveis valores de p são $p = 1$ e $p = 2$.

O corte CG-Nível (2.21) corresponde ao corte de Gomory para $p = 1$ e $d \geq 1$. Para tentar obter a faceta mencionada com estes parâmetros, faz-se $y = a_0^1 = 10$ na Proposição 2.5. Portanto, $\alpha = 15$, $\delta = 0$, e as condições (i) e (ii) implicam que $26/16 \leq d < 26/15$. O corte mais forte é obtido para o maior valor de d, como mostra a Proposição 2.4, e então, $d = 26/15 - \epsilon$. Assim, a Proposição 2.3 pode ser aplicada no cálculo dos coeficientes de x_1 e x_2. Note que a Proposição 2.6 deve ser utilizada para o cálculo de r e do termo $\lceil a/d \rceil$. Desta forma, temos o corte $z + x_1 + 2x_2 = 10$. Note na Figura 2.6 que este corte é dominado pela própria equação original do problema.

Entretanto, para $p = 2$, a faceta $z + x_1 + 3x_2 = 10$ pode ser encontrada. A partir da Proposição 2.5, $a_0^2 = 10$ implica $26/8 \leq d < 25/7$. Para $d = 25/7 - \epsilon$, aplicando novamente as proposições 2.3 e 2.6 e obtém-se a faceta $z + x_1 + 3x_2 = 10$, como mostra a Figura 2.6.

Para comparar o corte obtido com o corte GIF (1.14) utiliza-se um valor de d tal que $26/8 \leq d < 25/7$, como no corte CG-Nível.

Para $d = 7/2$, o corte GIF é $(8/3)s + x_1 \geq 4$. Colocando este corte em função das variáveis x_1 e x_2, temos $3x_1 + 8x_2 \leq 28$.

Observe na Figura 2.6 que, apesar de não haver relação de dominância entre o corte GIF e o corte CG-Nível (2.30), o corte GIF não constitui faceta. □

Exemplo 2.4. Este exemplo ilustra a aplicação do algoritmo de duas fases ALGDMaxmin, que maximiza os coeficientes positivos na primeira fase e aplica a segunda fase objetivando maximizar os negativos. Suponha o seguinte problema inteiro

$$\begin{array}{ll} \max & x_1 + x_2 \\ \text{s.a.} & -4x_1 + 5x_2 \leq 9, \\ & x_1 \leq 1, \\ & x \in Z_+^2. \end{array}$$

O corte CG-Nível é calculado para a primeira desigualdade para $p = 3$. Pela Proposição 2.4, o lado direito do corte, $y = a_0^3$ está limitado a $0 \leq y \leq 6$. Escolhe-se o valor $y = 3$.

Para obter o valor do parâmetro d que maximiza os coeficientes positivos, aplica-se o algoritmo ALGDMax. A Proposição 2.5 implica que $\alpha = 1$, $\delta = 0$, e, portanto, $d_{max} = p + (y+1)/\alpha - \epsilon = 7 - \epsilon$.

Suponha que não seja aplicada a segunda fase. Então prossegue-se com o cálculo dos demais coeficientes com o valor de d encontrado. Para calcular o coeficiente das variáveis x_1 e x_2 aplica-se a Proposição 2.3, observando-se que $d_{max} = d^* - \epsilon$, e portanto, a Proposição 2.6 deve ser utilizada.

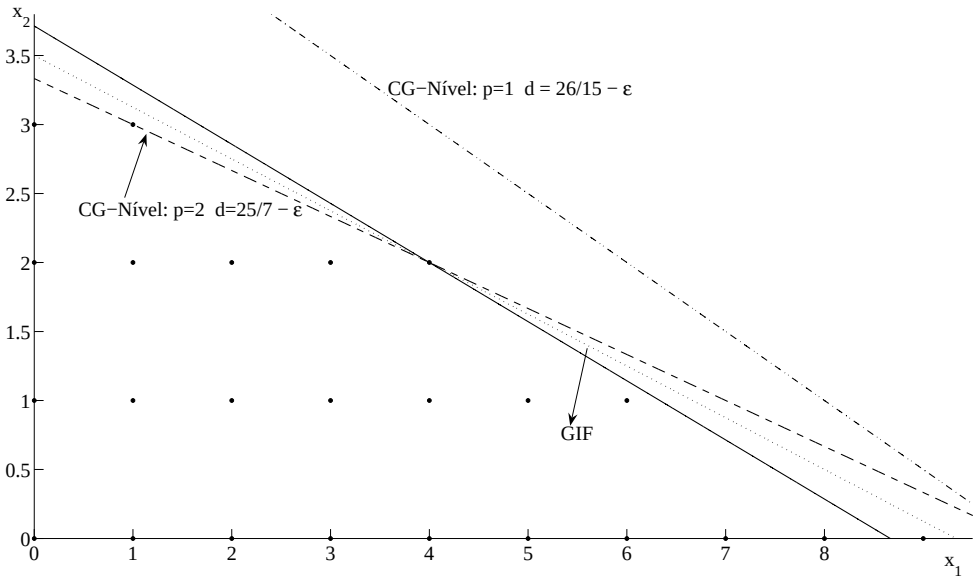

Figura 2.6: Desigualdades CG-Nível e corte GIF

Assim, para x_1, $a = -4$ e $d = 7 - \epsilon$ implicam que $r = \lceil a - d^* \left(\lceil a/d^* \rceil - 1 \right) \rceil = 3$. Portanto, $p = 3 \leq r$ e $a_1^3 = a - p \lceil a/d^* \rceil = -4$.

Analogamente, o coeficiente da variável x_2 pode ser calculado, obtendo-se $a_2^3 = 2$. Portanto, temos o corte

$$-4x_1 + 2x_2 \leq 3.$$

Suponha que a segunda fase seja aplicada. O algoritmo ALDMaxmin determina o intervalo válido de d para que o lado direito e o coeficiente de x_2 se mantenham inalterados. Pela Proposição 2.5, para que $a_2^3 = 2$ e $a_0^3 = 3$, devemos ter

$$a_0^3 = 3 \Rightarrow 4,5 \leq d \leq 7 - \epsilon$$
$$a_2^3 = 2 \Rightarrow d \geq 5$$

Portanto, $D_{inf} = 5$ mantém os coeficientes positivos inalterados. Utilizando este valor de d para recalcular o coeficiente negativo, temos $a_3^1 = -2$, e o novo corte

$$-2x_1 + 2x_2 \leq 3.$$

A Figura 2.7 mostra os dois cortes no R^2. Note que o corte calculado na segunda fase domina o primeiro. □

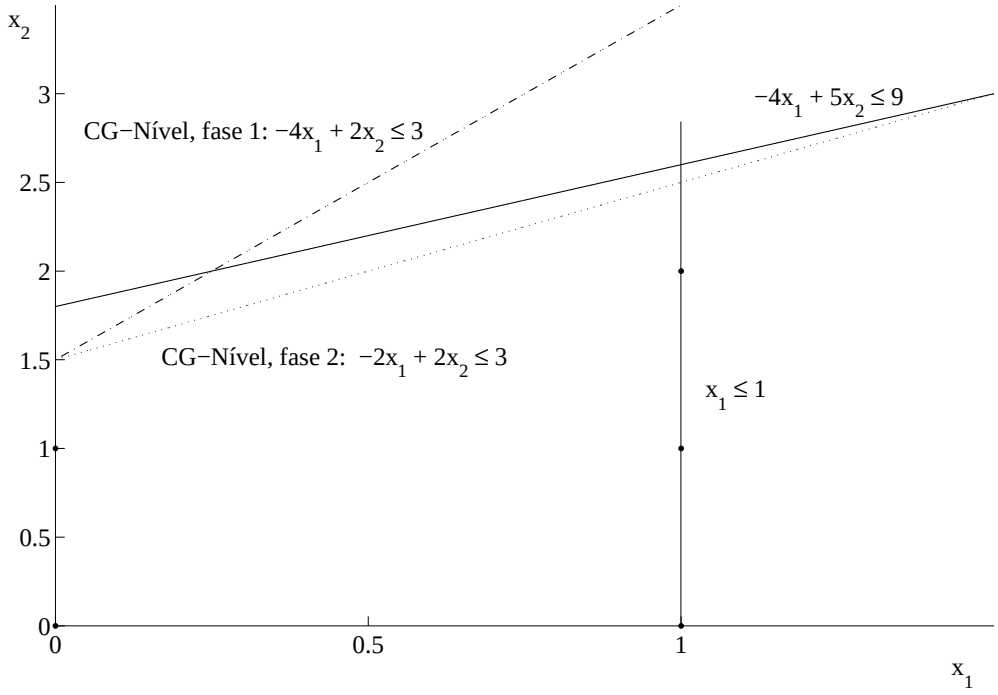

Figura 2.7: Aplicação do algoritmo de 2 fases.

2.5 Exemplo Ilustrativo para os Três Cortes

Exemplo 2.5. Considere o mesmo PI do exemplo 1.4. Renomeando-se as variáveis s_1 e s_2, a primeira restrição do tableau corresponde à relaxação do nó raiz deste exemplo é

$$x_1 + \frac{2}{7}x_3 - \frac{1}{7}x_4 = 2\frac{3}{7}.$$

Portanto, $f_0 = 3/7$, $f_3 = 2/7$, $f_4 = 6/7$. Para o corte CFF (2.11),

$$k = \lceil 1/f_0 \rceil - 1 = 2.$$

Assim, temos $p = \{1, 2\}$, e $J_0 = \{i \in J : f_i \in \left[0, \frac{3}{7}\right]\}$, $J_1 = \{i \in J : f_i \in \left(\frac{3}{7}, \frac{5}{7}\right]\}$ e $J_2 = \{i \in J : f_i \in \left(\frac{5}{7}, 1\right]\}$. Os conjuntos não vazios são, portanto, $J_0 = \{3\}$ e $J_2 = \{4\}$. A partir destes conjuntos calcula-se o corte CFF, $6x_3 + 4x_4 \geq 9$, que pode ser colocado em função das variáveis originais, gerando o corte

$$2x_1 + 2x_2 \leq 11.$$

Um novo corte CFF pode ser gerado a partir da aplicação do procedimento de multiplicação por -1 da linha do tableau, como descrito pelos autores no artigo original. Neste caso, temos a equação

$$-x_1 - \frac{2}{7}x_3 + \frac{1}{7}x_4 = -2\frac{3}{7}.$$

Note que $f_0 = 4/7$, $f_3 = 5/7$, $f_4 = 1/7$. A parte fracionária do lado direito tornou-se $> 1/2$, e portanto,

$$k = \lceil 7/4 \rceil - 1 = 1.$$

Assim, tem-se apenas os dois conjuntos $J_0 = \{j \in J : f_j \in \left[0, \frac{4}{7}\right]\}$ e $J_1 = \{j \in J : f_j \in \left(\frac{4}{7}, 1\right]\}$, ou seja, $J_0 = \{4\}$ e $J_1 = \{3\}$. Portanto, o corte CFF é a faceta

$$x_1 + x_2 \leq 5.$$

A Figura 2.8 mostra as desigualdades no R^2. Note que, neste exemplo, o corte calculado a partir da pré-multiplicação domina o corte obtido a partir do procedimento padrão de derivação.

Para a mesma linha tableau é derivado o k-corte para $k = 2$. Para este valor de k, a nova linha do tableau é dada por

$$2x_1 + \frac{4}{7}x_3 - \frac{2}{7}x_4 = 4\frac{6}{7}.$$

O corte (2.14) para esta equação é dado por $4x_3 + 5x_4 \geq 6$. Substituindo-se as variáveis de folga pelas variáveis originais, temos

$$x_1 + 2x_2 \leq 9.$$

Observe na Figura 2.9, que o corte GIF consegue limitar de forma mais eficiente a região factível neste exemplo.

Considere, agora, o corte CG-Nível (2.3) para $p = 2$, calculado sobre a primeira restrição do modelo, $3x_1 + x_2 \leq 11$. Note que, pela Proposição 2.4, o coeficiente do corte correspondente à variável x_2 está fixo em 0 para qualquer valor de d.

Para encontrar a faceta $x_1 \leq 3$, supõe-se $y = a_1^2 = 1$, e a Proposição 2.5 é aplicada. Desta forma, temos

$$\alpha = \lceil (a - y)/p \rceil - 1 = \lceil (3 - 1)/2 \rceil - 1 = 0.$$

Analogamente, $\delta = 0$. Portanto a condição (i) implica que $d \geq 3$. Supondo, agora, $a_0^2 = 3$, temos $\alpha = 3$ e $\delta = 0$. A condição (i) impõe que $d \geq 11/4$, enquanto que a condição (ii) impõe $d < 10/3$. Portanto, para $d = 3$ temos a faceta $x_1 \leq 3$, representada na Figura 2.9.

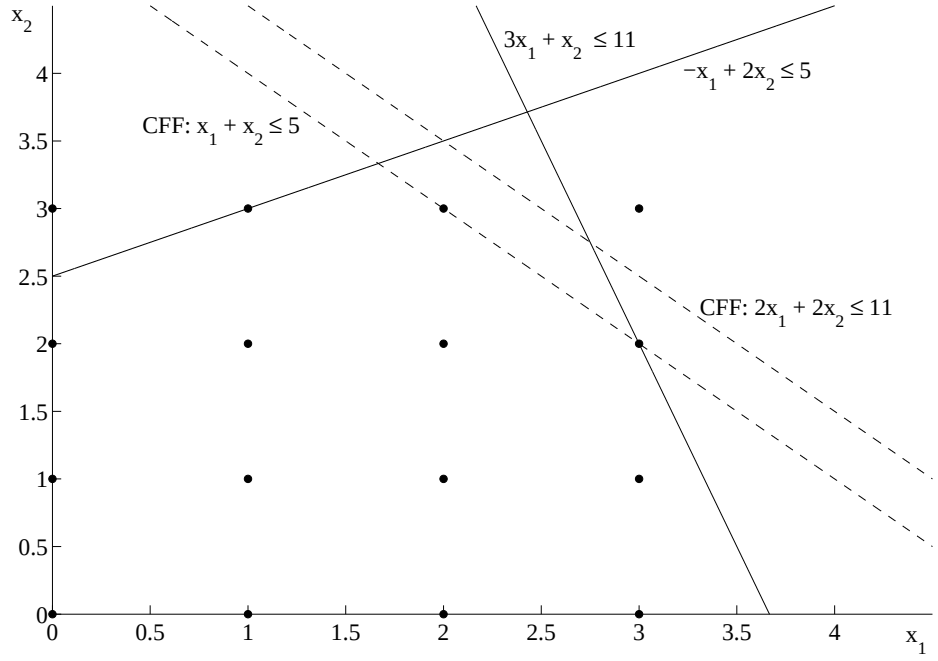

Figura 2.8: Comparação entre cortes CFF com e sem pré-multiplicação por -1.

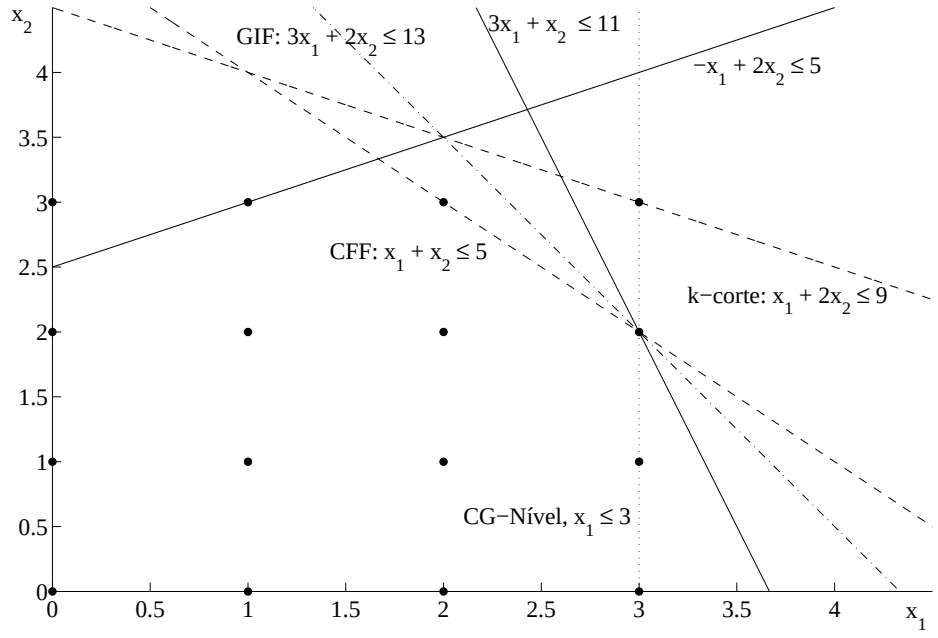

Figura 2.9: Comparação entre cortes CFF, CG-Nível e K-cortes.

Capítulo 3

Implementação Computacional e Resultados

Este capítulo descreve os testes computacionais propostos para o Corte Fracionário Forte (CFF), K-corte e CG-Nível. É feita uma análise sobre a melhor estratégia de inserção de cortes no contexto do método *branch-and-cut*, utilizando a biblioteca do software CPLEX 10.1. Para cada corte são testados diferentes valores de seus respectivos parâmetros. Resultados comparativos mostram o desempenho de cada corte em instâncias de problemas da mochila multidimensional, problemas de designação generalizada e problemas inteiros da MIPLIB.

3.1 Método *branch-and-cut*

Uma descrição geral do método *branch-and-cut* foi apresentada na Seção 1.4. Este capítulo objetiva detalhar a implementação computacional deste trabalho, baseada no software CPLEX 10.1 (ILOG, 2006a). Um maior detalhamento do uso da biblioteca do CPLEX é feito na Seção 3.4.

A implementação computacional envolve decisões a respeito de como os cortes são usados no contexto do método *branch-and-cut*. Na Seção 3.2 são descritos os parâmetros testados para características inerentes ao método. Em um determinando nó da árvore, é feito um *round* de cortes, que consiste em gerar cortes a partir de todas as linhas do tableau com lado direito fracionário. Portanto, define-se o parâmetro *frac_rhs* que determina a que distância mínima de um valor inteiro deve estar o lado direito de uma linha do tableau para que esta seja utilizada na geração de cortes. O tamanho do *round*, *max_round_perc*, é o parâmetro que define a porcentagem dos cortes gerados a partir das linhas com lado direito fracionário que são inseridos no PL.

Critérios de descarte de corte são aplicados a este *round* gerado com o objetivo de evitar a inserção de cortes que não violem a solução atual ou introduzam instabilidade numérica ao PL. Em seguida é aplicado o critério de eficiência ou de violação normalizada, definidos na Seção 1.5 para ordenação e escolha dos cortes a serem inseridos. A relaxação do lado direito do corte antes de sua inserção também é considerada através do parâmetro *rhs_relax*.

As instâncias utilizadas para testes e o ambiente computacional utilizado estão descritos nas seções 3.3 e 3.4, respectivamente.

A Seção 3.5 apresenta testes comparativos de duas estratégias de inserção de cortes nos nós. Na primeira, o *round* de cortes é aplicado segundo o procedimento descrito acima. Na segunda, é feita reotimização do PL após a inserção dos cortes e são removidos aqueles que possuírem variável de

folga básica. São apresentados também testes relativos aos parâmetros definidos na Seção 3.2, como tamanho do *round*, uso de descarte por paralelismo, e critério de seleção de cortes. Ao fim desta seção são descritos os testes relativos aos parâmetros de cada plano de corte.

Na Seção 3.6 são descritos os testes comparativos dos três planos de corte segundo os melhores parâmetros encontrados.

3.2 Geração de Cortes a partir do Tableau

A cada nó da árvore de enumeração é necessário definir que linhas do tableau são usadas para gerar cortes e o número de cortes a que cada linha dará origem (número de *rounds*). Como já colocado anteriormente, um *round* é definido como a geração de um corte para cada linha tableau cuja variável básica é inteira e tem valor fracionário.

Em (Letchford e Lodi, 2002), são gerados no nó raiz da árvore, 1, 10 e 25 *rounds* de cortes para cada um dos cortes implementados. Goycoolea (2006) observa que a maioria dos softwares comerciais adicionam no máximo dois *rounds* de cortes do tipo GIM ao mesmo nó. Para efetuar a comparação entre os cortes neste trabalho, é gerado um *round* de cortes a cada nó da árvore, e o tamanho deste *round* é um parâmetro testado.

Lado Direito Fracionário

Apenas as linhas do tableau com variável básica inteira cujo valor é fracionário são usadas para geração de cortes. Apesar de considerar-se problemas de programação inteira, variáveis reais de folga são inseridas ao problema com a aplicação de cortes GIM, por exemplo.

Na biblioteca COIN-OR (http://www.coin-or.org), fica definido para os cortes de Gomory e *Reduce-and-Split*, que qualquer número que se afaste mais que "away" de um numero inteiro é considerado fracionário. O valor deste parâmetro é $0{,}05$ e pode ser alterado. Define-se abaixo o parâmetro *frac_rhs*, que tem a função de descartar linhas cuja variável básica tenha valor praticamente inteiro.

- **Parte fracionária mínima (*frac_rhs*):** Este parâmetro define o mínimo e o máximo valor que deve ter a parte fracionária do lado direito para que seja aplicado o corte. A principal motivação é não gerar cortes sobre linhas cujo lado direito seja inteiro. Assim, dado um lado direito do tableau, $\bar{a}_0$, devemos ter

  $$\textit{frac\_rhs} \leq \bar{a}_0 - \lfloor \bar{a}_0 \rfloor \leq 1 - \textit{frac\_rhs}.$$

Tamanho do Round

A inserção de um determinado número fixo de cortes pode estar dividida em *rounds* parciais de tamanhos diferentes. Em (Balas et al., 1996a) é ressaltado que, em geral, o tempo total para se inserir um dado número cortes é maior quanto menor o tamanho do *round*. Isto porque, entre cada *round* parcial deve haver reotimização do novo PL. É feita uma comparação em relação ao melhoramento do gap de integralidade em um PL quando se varia o tamanho do *round*. Para tanto, são gerados inicialmente 5 *rounds* completos de cortes, e o tempo total é armazenado. Em seguida, é feito um teste em que o mesmo tempo total é dispensado na geração de consecutivos *rounds* parciais: são

calculados cortes para as linhas cuja variável básica têm parte fracionária mais próxima de $1/2$. Os *rounds* parciais testados pelos autores têm tamanho 50% e 10% do total de linhas.

Os resultados demonstram que nenhuma das três abordagens é melhor para todas as instâncias. Entretanto, o gap de integralidade é menor para a primeira estratégia em 9 das 16 instâncias testadas. Baseados nestes experimentos, os autores definem o parâmetro MAXROUND. Desta forma, o tamanho do *round* é o mínimo entre o número de linhas com variável básica fracionária e MAXROUND. Na implementação, o tamanho máximo do *round* é 40 para instâncias com até 400 variáveis binárias, e 80 para problemas com mais variáveis.

Além disso, quanto maior o número de cortes inseridos, maior a chance de gerar erros de precisão durante a reotimização. Portanto, propõe-se a geração do *round* completo de cortes e a seleção do subconjunto de melhor qualidade (Goycoolea, 2006), como definido a seguir.

- **Tamanho máximo do *round* (*max_round_perc*):** Para todas as linhas que não foram descartadas pelo critério *frac_rhs*, o corte é calculado. Seja C este número de cortes. O máximo de cortes a serem inseridos é $I = \lceil \textit{max\_round\_perc} \times C \rceil$. São aplicados critérios de descarte aos C cortes, restando C'. Os C' cortes são ordenados por valores decrescentes de violação normalizada ou eficiência, e são inseridos o mínimo entre I e C' cortes.

Seleção de Cortes Gerados

Quando apenas um subconjunto dos cortes gerados em um nó será inserido, é necessário definir um critério de classificação dos cortes. Assim como foi observado no Capítulo 1, dois critérios são comumente utilizados na literatura.

- **Critério de seleção de cortes (*crit_sel*):** Define qual critério de classificação de cortes é utilizado: eficiência ou violação normalizada.

Descarte de Cortes Gerados

Após gerado o *round* completo de cortes, é necessário analisar a qualidade de cada corte gerado, de modo a descartar aqueles que pouco violam a solução atual ou tendem a gerar instabilidade numérica. Assim, cortes que forem incluídos em algum dos critérios abaixo não são inseridos no PL.

- **Violação mínima (*viol_min*):** Violação mínima requerida para que o corte seja inserido no PL.

Cornuéjols (2008) explicita a necessidade de evitar erros por instabilidade numérica através do descarte de cortes cuja razão entre o maior e o menor coeficiente é muito grande. Em (Goycoolea, 2006), este parâmetro fica definido a priori como 10^5.

- **Razão máxima entre coeficientes (*max_razao*):** Razão máxima entre o valor absoluto do maior e do menor coeficiente do corte.

A inserção de um corte muito denso em um determinado nó também é uma fonte de erros numéricos (Cornuéjols, 2008). Durante a reotimização, os coeficientes do novo corte se propagam pela inversa da base, e estas operações acarretam erros. Goycoolea (2006) define um valor constante para o número máximo de coeficientes diferentes de zero na linha do corte (máximo de 500).

Entretanto, as instâncias aqui utilizadas são bastante heterogêneas, e, portanto, a limitação máxima deve estar relacionada ao tamanho da instância.

- **Número máximo de coeficientes (*max_nao_nulos*):** Percentual máximo de coeficientes diferentes de zero na linha do corte.

A adição de cortes semelhantes ao mesmo problema linear deve ser evitada através da definição de um critério de eliminação de cortes semelhantes. O paralelismo entre dois cortes está definido em (1.23). O parâmetro *max_par* $\in [0, 1)$, é o máximo valor para este parâmetro.

- **Paralelimo entre cortes (*max_par*):** Máximo paralelismo entre dois cortes. Um corte $\alpha x \leq \alpha_0$ é descartado quando tem uma eficiência menor que outro corte $\beta x \leq \beta_0$, tal que par $(\alpha, \beta) >$ *max_par*.

Relaxação de Cortes Inseridos

Para evitar que soluções inteiras sejam descartadas por erros numéricos, é comumente utilizado na literatura a adição de uma "folga" ao lado direito do corte. Andersen et al. (2005) utilizam a subtração de um valor ϵ para enfraquecer os seus cortes Reduce-and-Split para PIM. Dada uma desigualdade do tipo $\alpha x \geq \alpha_0$, a nova desigualdade enfraquecida é dada por $\alpha x \geq \alpha_0 - \epsilon$. Os autores utilizam um valor fixo 10^{-10}. Este valor mostra-se suficiente para garantir que a solução ótima não seja excluída. Seus cortes são testados utilizando-se o CPLEX, versão 8.0 e aplicando o método *cut-and-branch*.

Na biblioteca COIN-OR, a mesma técnica é utilizada para a implementação dos cortes de Gomory (GIM) e Reduce-and-Split. No segundo caso, ao final da geração do corte é aplicada uma correção constante, cujo valor padrão é 10^{-8} e pode ser alterado. Também no corte GIM o lado direito é relaxado em um valor fixo (10^{-8}) para cortes com poucos elementos. Entretanto, quando o número de elementos excede uma determinada constante, a correção é proporcional ao número de elementos da linha.

Baseando-se nas observações anteriores, define-se o parâmetro *rhs_relax*, ou relaxação do lado direito, que é um valor constante adicionado ao lado direito da equação do corte, ao final de sua geração.

- **Relaxação do lado direito (*rhs_relax*):** Relaxação adicionada ao lado direito do corte. Todos os corte gerados são desigualdades do tipo "$\leq$". Sendo assim, para um corte $\alpha x \leq \alpha_0$, temos o novo corte dado por $\alpha x \leq \alpha_0 +$ *rhs_relax*.

Critérios de Parada

A implementação visa prioritariamente a comparação entre os diversos cortes e cada um deles necessita de intervalos de tempo diferentes para ser calculado. Os cortes CG-Nível e K-corte, por exemplo, devem gerar, para cada linha, cortes para diversos valores de seus respectivos parâmetros, inserindo apenas o melhor entre eles. Portanto, levam mais tempo na geração de cortes que os demais.

Além disso, os critérios de eliminação de cortes podem atingir mais a um corte que a outro, e o tamanho final do *round* vai variar para cada algoritmo. Como resultado, um número diferente de cortes é inserido para cada algoritmo.

Assim, fica claro que uma comparação por número de nós não seria justa. Como critério de parada, usamos, portanto, o tempo máximo de execução ou até que a solução ótima seja encontrada.

3.3 Instâncias

3.3.1 Mochila Muldimensional

O problema da mochila muldimensional (*multidimensional knapsack problem*) é um problema NP-completo tratado por diferentes abordagens (exatas e heurísticas) na literatura.

O problema da mochila clássico consiste em armazenar n itens em uma mochila de capacidade b. A cada item j está associado um "lucro" p_j. No problema da mochila multimensional, são consideradas m mochilas, e a inserção do item x_j deve respeitar a capacidade de cada uma elas. Desta forma, temos a formulação abaixo.

$$
\begin{aligned}
(MKP) \qquad \text{max} \qquad & \sum_{j=1}^{n} p_j x_j \\
\text{sujeito a} \qquad & \sum_{j=1}^{n} r_{ij} x_j \leq b_i, \quad i = 1, ..., m \\
& x_j \in \{0, 1\}, \quad j = 1, .., n.
\end{aligned}
$$

As instâncias utilizadas foram retiradas da OR-Library (Beasley, 1990), e foram originalmente propostas por Chu e Beasley (1998). Os problemas têm número de restrições $m = \{5, 10, 30\}$. O número de itens varia entre os valores $n = \{100, 250, 500\}$. Para cada par (m, n), são geradas 30 instâncias, totalizando 270.

As 30 instâncias são separadas em 3 grupos, de acordo com a forma como as restrições são geradas. Para cada par (m, n), o lado direitos das restrições de mochila são gerados a partir da equação

$$b_i = \alpha \sum_{j=1}^{n} r_{ij},$$

onde $\alpha = 0{,}25$ para os primeiras 10 instâncias, $\alpha = 0{,}5$ para os próximas 10 instâncias e $\alpha = 0{,}75$ para as demais. Os coeficientes r_{ij} foram gerados segundo uma distribuição uniforme discreta variando entre 0 e 1000, e os coeficientes p_j são correlacionados a r_{ij}.

Note que, quanto menor o valor de α, mais forte se torna a restrição resultante. Os resultados computacionais de Chu e Beasley (1998) confirmam uma maior dificuldade no fechamento do gap para estas instâncias.

As instâncias são referenciadas como a seguir. A instância *mkp5.100-17* corresponde à décima sétima instância de mochila multidimensional com 5 restrições de mochila e 100 itens, ou seja, $(m, n) = (5, 100)$. A melhor solução inteira encontrada na literatura está disponível na OR-Library.

3.3.2 Mochila Multidimensional com Restrições de Demanda

O problema da mochila multidimensional com restrições de demanda (*multi-demand multidimensional knapsack problem*) é uma extensão do problema descrito na seção anterior em que são

adicionadas q restrições de demanda além das restrições de mochila. A formulação deste problema é

$$
\begin{aligned}
(MDMKP) \qquad \text{max} \quad & \sum_{j=1}^{n} p_j x_j \\
\text{sujeito a} \quad & \sum_{j=1}^{n} r_{ij} x_j \leq b_i, \quad i = 1, ..., m \\
& \sum_{j=1}^{n} r_{ij} x_j \geq b_i, \quad i = m+1, ..., m+q \\
& x_j \in \{0, 1\}, \quad j = 1, .., n.
\end{aligned}
$$

As instâncias deste problema foram retiradas da OR-Library (Beasley, 1990) e são modificações das instâncias de mochila multidimensional descritas na seção anterior. Dada uma instância da mochila multidimensional com m restrições do tipo $\leq$, são criadas 6 novas instâncias que correspondem à combinações dos valores de $q = 1$, $q = m/2$ e $q = m$ com valores de função objetivo positivos ou mistos.

As instâncias recebem o nome $m - n - q - r - s$, em que m é o número de restrições de mochila, n é o número de variáveis, q é o número de restrições de demanda e $r = 1$ se há coeficientes negativos na função objetivo. O índice $s = 0, 1, ..., 14$ é o número que identifica a instância dentro do grupo $m - n - q - r$ (Arntzen et al., 2006).

3.3.3 Problema de Designação Generalizada

O problema de designação generalizada (*generalized assignment problem*) consiste em associar m agentes a n tarefas, de forma que cada tarefa seja designada a exatamente um agente. Para tanto, define-se a variável binária x_{ij} que é igual a 1 caso ocorra a designação do agente i à tarefa j, e 0, caso contrário. As restrições consistem em respeitar a capacidade do agente, b_i, dado que cada tarefa j consome uma quantidade de recursos r_{ij} do agente i. À designação do agente i à tarefa j está associado um custo c_{ij}. O objetivo é minimizar o custo total de designação.

$$
\begin{aligned}
(GAP) \qquad \text{min} \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{sujeito a} \quad & \sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, ..., n \\
& \sum_{j=1}^{n} r_{ij} x_{ij} \leq b_i, \quad i = 1, .., m \\
& x_{ij} \in \{0, 1\}, \quad i = 1, .., m, \quad j = 1, .., n.
\end{aligned}
$$

O primeiro conjunto de instâncias foi retirado da OR-Library (Beasley, 1990) e é constituído de 12 grupos de 5 instâncias. Cada grupo corresponde a um par (m, n). A instância *c515-1* corresponde

ao problema número 1 das instâncias com 5 agentes e 15 tarefas, enquanto a instância *c1060-5* corresponde à instância de número 5 com 10 agentes e 60 tarefas. As instâncias citadas correspondem, respectivamente, à instância com o menor e o maior número de variáveis deste conjunto. A solução ótima de cada instância está disponível na OR-Library.

As demais instâncias estão divididas em 5 grupos: A, B, C, D e E. Estão disponíveis na OR-Library, 6 instâncias dos grupos A,B,C e D, com $n \leq 200$. Estas instâncias também estão disponíveis através do endereço eletrônico *http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/gap*. Neste endereço, além de outras 6 instâncias com $n \leq 200$, pertencentes ao grupo E, estão disponíveis instâncias maiores, geradas por Yagiura et al. (2006), para os grupos C, D e E. São 9 instâncias para cada grupo, com $n \leq 1600$. Neste mesmo trabalho os autores disponibilizam a solução ótima ou melhor solução encontrada para instâncias do tipo B,C,D e E. Instâncias do grupo A são de resolução mais fácil e a solução ótima pode ser encontrada em um tempo computacional pequeno.

A notação adotada para as instâncias é simples. A instância *b20200*, por exemplo, corresponde a uma instância do tipo B, com 20 agentes e 200 tarefas.

3.3.4 MIPLIB 3.0 e MIPLIB 2003

A biblioteca eletrônica de problemas inteiros e problemas inteiros mistos, MIPLIB, pode ser encontrada em duas versões. A primeira delas foi disponibilizada eletronicamente em 1996 (Bixby et al., 1996) e é conhecida como MIPLIB 3.0. A segunda corresponde à quarta versão e ficou conhecida como MIPLIB 2003 (Achterberg et al., 2003). Foram selecionadas para testes as instâncias de programação inteira pura presentes nas duas bibliotecas.

3.4 CPLEX 10.1

A implementação computacional utilizou a linguagem C e a biblioteca CPLEX denominada *callable library*. Ao contrário da outra biblioteca disponível (*Concert Technology*, para C++), esta biblioteca permite que o usuário acesse diretamente as funções avançadas do CPLEX, que incluem, por exemplo, aquelas necessárias para a recuperação das linhas do tableau.

Os algoritmos de planos de corte foram implementados no software CPLEX versão 9.0, e posteriormente portados para o CPLEX versão 10.1 (ILOG, 2006a). Dentro do subconjunto de funções utilizadas, a biblioteca do CPLEX é a mesma para estas duas versões do software. Como uma das vantagens, a nova versão traz um novo parâmetro, *numericalemphasis*, que aumenta a precisão dos cálculos internos (ILOG, 2006b).

O método *branch-and-cut* foi utilizado segundo as configurações padrões. Entretanto, o pacote CPLEX pode aplicar, por padrão, uma variedade de planos de corte durante a resolução dos problemas. Estão inclusos os cortes *clique, covers, disjunctive cuts, flow cover cuts, flow path cuts, gomory fractional cuts, gub covers, implied cuts, e MIR cuts*. Para evitar possíveis interações indesejadas com os cortes aqui implementados, todos estes cortes nativos do CPLEX foram desabilitados, assim como as heurísticas. O pré-processamento também foi totalmente desabilitado.

3.4.1 Inserção de Cortes

O software CPLEX provê várias formas de se adicionar cortes ao modelo. Em todas elas, o usuário é responsável por desenvolver rotinas que são usadas convenientemente pelo CPLEX ao longo da execução do método *branch-and-cut*. Tais funções são conhecidas como *callbacks*.

A forma mais simples de adicionar cortes é através da função *CPXaddusercuts*, que adiciona cortes globais, e pode ser chamada antes que se inicie a resolução do problema. Contudo, além da limitação de só inserir cortes globais, esta função não pode ser usada ao longo do *branch-and-bound*. A função *CPXcutcallbackaddlocal*, por outro lado, permite que sejam adicionados cortes locais durante o processo de *branch-and-bound*. Da mesma forma, a rotina *CPXcutcallbackaddglobal* permite a inserção de cortes com validade global.

Além da adição de cortes, também pode ser desejada a remoção local daqueles não ativos. Neste caso, o PL do nó atual é duplicado e os cortes gerados são inseridos nesta cópia. O novo PL é resolvido, e então somente aqueles cortes que não possuírem sua respectiva variável de folga básica são inseridos no PL original do nó.

3.4.2 Plataforma de Testes

Todos os resultados apresentados nas seções a seguir foram obtidos a partir da execução do CPLEX 10.1 em uma máquina Intel Pentium 4, com *clock* de 2,80GHz, e 512Mbytes de memória RAM. O código foi compilado e executado no sistema operacional Red Hat Linux 3.3.3-7, usando o compilador GCC versão 3.3.3, com a opção "-O2". As medidas de tempo foram calculadas a partir de chamadas do sistema operacional que contabilizam apenas o tempo de processamento do processo (descontando atrasos por acesso a disco ou tempo gasto por outros processos, por exemplo).

3.5 Resultados Computacionais

Os testes computacionais envolvem a comparação dos cortes fracionários, corte de Gomory para programação inteira mista (GIM), Corte Fracionário Forte (CFF), k-cortes e cortes CG-Nível. Apesar de serem usadas para testes apenas instâncias com variáveis binárias ou inteiras, os cortes GIM devem ser utilizados em detrimento dos cortes GIF porque as variáveis de folga adicionadas ao problema pela inserção destes cortes são reais. O mesmo é válido para os k-cortes.

Os testes computacionais envolvem o cálculo do percentual do gap de integralidade fechado ao final da execução. Seja PI um problema inteiro de minimização, P a sua corresponde relaxação linear e P_1 a relaxação correspondente ao limitante inferior da árvore de *branch-and-cut*. Assim, o percentual de gap fechado é dado por

$$\% \text{ Gap fechado} = 100 \times \frac{\text{ótimo}(P_1) - \text{ótimo}(P)}{\text{ótimo}(PI) - \text{ótimo}(P)}$$

3.5.1 Uso de Reotimização

Este teste tem o objetivo de comparar duas técnicas de inserção de cortes. A diferença entre elas é o uso ou não de reotimização após a inserção de cortes. Na primeira, chamada inserção completa

(IC), os cortes são inseridos localmente nos nós da árvore e segue-se o procedimento padrão de reotimização do PL e continuação do *branch-and-cut*.

A segunda estratégia baseia-se na criação de um PL adicional onde é feita a inserção de cortes. É aplicada a reotimização deste PL e são adicionados ao PL original apenas os cortes que não apresentam variável de folga básica. Este procedimento é usado em (Balas et al., 1996b). Esta técnica de inserção com reotimização e remoção dos cortes é aqui denominada IR.

Foram utilizadas para testes as instâncias c515-1, c515-2, b05100 e b05200, correspondentes a problemas de designação generalizada, as instâncias p0033, p0201, da MIPLIB 3.0 e as instâncias de mochila multidimensional mkp5.100-00 e mkp5.100-01. A Tabela 3.1 contém os valores dos parâmetros utilizados. A Tabela 3.2 mostra os valores médios obtidos para este conjunto de instâncias com um tempo computacional máximo de 1200 segundos.

Parâmetro	Valor
frac_rhs	0,05
max_round_percent	100%
viol_min	10^{-4}
max_razao	10^{5}
max_nao_nulos	100%
max_par	100%
rhs_relax	0
crit_sel	eficiência

Tabela 3.1: Parâmetros usados no teste de comparação entre IC e IR

Note que o percentual de gap fechado é maior para todos os cortes quando é aplicada reotimização seguida de remoção de cortes com variável de folga básica. Apesar disso, a razão gap fechado por nó pode ser menor para esta técnica, devido ao maior número de nós avaliados. O tempo médio de execução é menor quando a remoção de cortes é efetuada, indicando que soluções ótimas são encontradas em um tempo menor para esta técnica. O tempo médio gasto por nó também é menor, pois a inserção excessiva de cortes acarretada pela técnica IC faz com que o problema linear de cada nó torne-se mais difícil de resolver. Note que o número de cortes gerados é maior para IR, pois mais nós são avaliados, mas um grande percentual destes cortes é removido.

As figuras a seguir ilustram o comportamento indicado para a instância b05100 e o corte GIM. A Figura 3.1 indica que o número de nós avaliados em um mesmo intervalo de tempo é maior quando são removidos cortes após a reotimização. A Figura 3.2 mostra o número de cortes em função do número de nós, e evidencia a maior inserção de cortes pela técnica IC.

A Figura 3.3 apresenta o percentual do gap de integralidade fechado para as duas técnicas. Note que o gap fechado é sempre maior para qualquer instante de tempo quando a técnica IR é utilizada, chegando a 100% antes de 400 segundos, enquanto o não uso de remoção implica no não fechamento total do gap.

Média	**Gomory**		**GIM**		**CFF**		**k-corte**		**CG-Nível**	
	IC	**IR**	**IC**	**IR**	**IC**	**IR**	**IC**	**IR**	**IC**	**IR**
Gap fechado(%)	20,63	74,54	79,66	94,43	33,56	75,06	83,05	88,40	22,67	70,53
Número de nós	38,25	7340,38	5078,00	10120,88	74,13	8387,13	7641,13	11404,00	77,00	6196,88
Cortes Gerados $(\times 10^3)$	9,9	140,1	19,3	35,2	7,5	138,8	14,9	22,8	5,5	99,1
Cortes Descartados		89%		73%		89%		61%		83%
Tempo(s)	1050,00	562,25	615,00	367,75	996,38	559,25	492,63	358,00	1050,25	629,13
Gap fechado por nó	0,45%	0,27%	0,37%	0,40%	0,35%	0,21%	0,35%	0,41%	0,32%	0,43%
Tempo por nó(s)	64,42	1,02	1,16	0,07	46,29	0,93	0,68	0,32	56,70	2,61

Tabela 3.2: Comparação entre as técnicas IC e IR

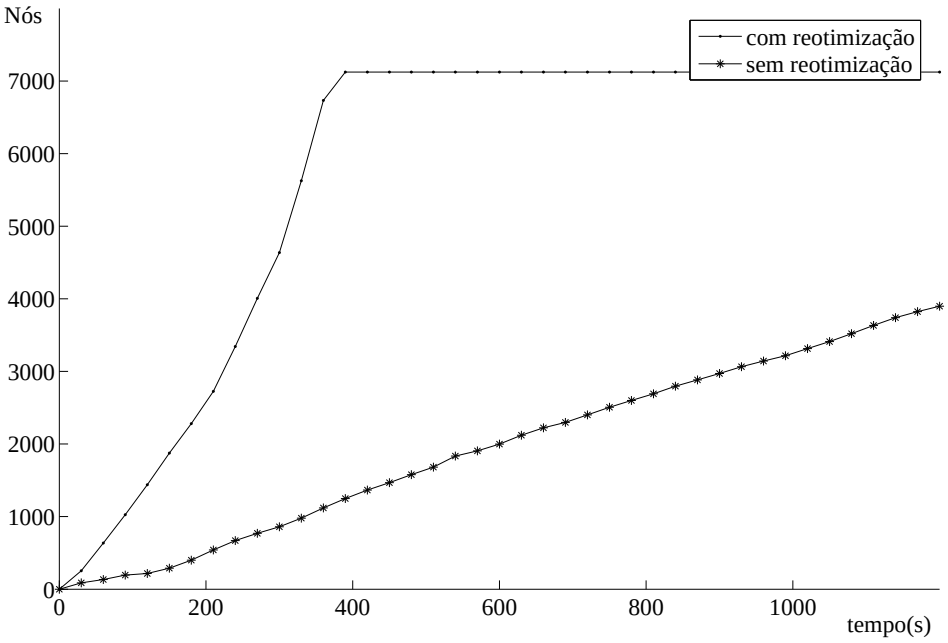

Figura 3.1: Número de nós para a instância b05100

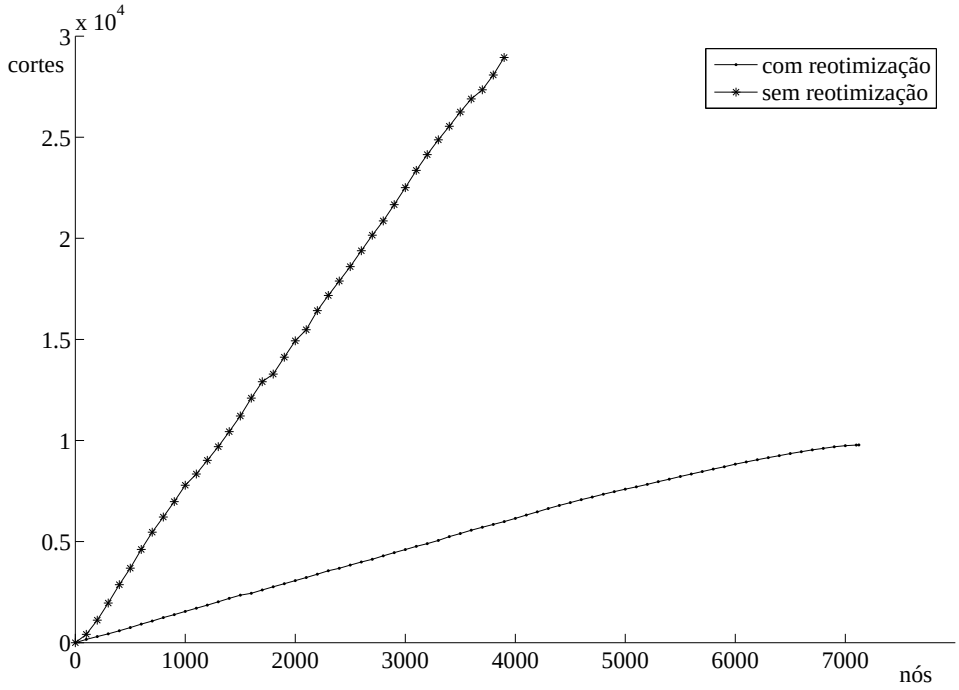

Figura 3.2: Número de cortes inseridos para a instância b05100

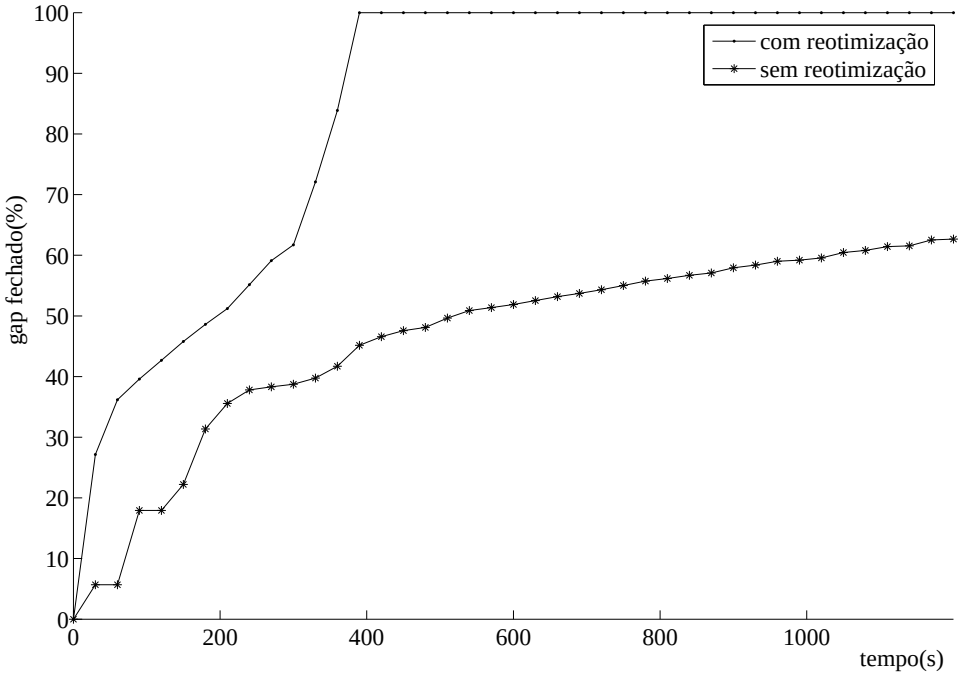

Figura 3.3: % Gap fechado para a instância b05100

3.5.2 Testes de Parâmetros

A seção anterior indica claramente dominância da técnica de inserção com reotimização (IR) em detrimento da inserção completa (IC). Portanto esta técnica é adotada nesta fase de ajuste de parâmetros e em todos os testes que se seguem. Para todos os parâmetros citados na Seção 3.2 é feito o ajuste de valores na presente seção, utilizando um subconjunto das instâncias disponíveis.

O parâmetro *frac_rhs*, que define a parte fracionária mínima que deve ter o lado direito da linha do tableau para que seja considerada na geração de cortes, foi mantido fixo em $0{,}05$ na seção anterior. Este valor mostrou-se adequado pois não foram gerados erros de precisão em nenhuma instância. Portanto, para os demais testes, este valor é mantido.

A relaxação do lado direito, *rsh_relax*, foi mantida em 0 para todos os cortes na seção anterior. Como nenhum erro de precisão foi gerado, este valor foi mantido nos demais testes.

Critérios de Descarte

Alguns dos critérios de descarte de corte também foram calibrados segundo os testes da seção anterior. O parâmetro *viol_min* define a mínima violação em relação à solução atual que deve ter um plano de corte para que considere-se que ele viola de fato a solução atual. O valor adotado para este parâmetro é 10^{-4}, assim como em (Goycoolea, 2006).

Da mesma forma, a razão máxima entre os coeficientes foi mantida em 10^5. O número máximo de coeficientes não nulos é fixado em 100%, ou seja, não é necessário limitar este coeficiente, pois utilizando-se seu valor máximo, nenhum erro de precisão foi gerado para nenhuma das instâncias dos

testes da seção anterior.

Instâncias

Para os testes de parâmetros das próximas seções foram consideradas as instâncias da mochila multidimensional, mkp10.100-00, mkp10.100-10, mkp10.250-00, mkp10.250-10, mkp10.250-20, instâncias da MIPLIB, harp2, air04, air05 e p2756, e instâncias de GAP, b05200, c10100 e d05200.

Tamanho Máximo do Round

Os parâmetros usados para este teste são os mesmos da Tabela 3.1, a não ser pelo parâmetro de paralelismo que é mantido em *max_par* $= 99\%$.

Foram considerados três valores para o tamanho máximo do *round*: 10%, 50% e 100%. A Tabela 3.3 mostra o gap de integralidade fechado para os cortes GIM, CFF e CG-Nível, em um tempo máximo de execução de 1800 segundos.

	GIM			CFF			CG-Nível		
Instância	10%	50%	100%	10%	50%	100%	10%	50%	100%
mkp10.100-00	58,96	59,37	57,91	49,65	44,48	43,75	54,48	52,57	53,07
mkp10.100-10	67,51	72,29	73,20	57,06	54,38	50,51	60,78	63,32	60,94
mkp10.250-00	27,88	28,16	25,75	21,02	20,56	20,52	24,50	22,29	24,76
mkp10.250-10	34,83	32,90	32,99	28,41	26,09	24,40	31,07	30,15	30,95
mkp10.250-20	37,86	34,04	35,78	22,34	23,27	30,76	31,62	31,75	30,48
air04	25,44	11,54	21,02	29,63	14,67	10,30	26,29	23,64	23,61
air05	58,69	34,42	20,97	47,67	34,93	31,69	48,05	32,67	31,12
harp2	53,69	68,57	55,34	70,38	61,46	63,90	68,53	33,86	42,87
p2756	3,07	3,55	3,55	2,13	2,95	2,13	3,24	2,13	2,27
b05200	31,72	44,60	40,42	54,62	26,48	52,40	100,00	43,14	49,88
c10100	66,05	61,80	68,80	56,24	54,42	45,36	41,62	47,91	21,86
d05200	14,38	15,33	16,50	2,48	5,90	6,69	9,55	12,20	12,13
Média	40,01	38,88	37,69	36,80	30,80	31,87	41,64	32,97	32,00

Tabela 3.3: % Gap fechado com variação tamanho do *round*

Note que para os três cortes a maior média é obtida para o tamanho do *round* 10%. O ganho no fechamento do gap para este tamanho de *round* é de aproximadamente 8% em relação aos demais, para o corte CG-Nível. Para os cortes CFF este ganho é de aproximadamente 5%. Para os cortes GIM, o ganho é menor e em torno de apenas 2% em relação ao *round* completo.

A Tabela 3.4 mostra as médias de nós avaliados, cortes gerados, e cortes inseridos após a reotimização. É possível notar a tendência de diminuição de nós avaliados quando o tamanho do *round* é aumentado, pois a adição de mais cortes aumenta o tempo gasto na reotimização. Este efeito da diminuição da quantidade de nós pode contrabalancear o ganho obtido pela inserção de mais cortes.

Nota-se que o número de cortes que são inseridos no PL após a reotimização é semelhante para os três tamanhos de *rounds*. Isto pode ser explicado pelo fato de que a utilização de um *round* maior

	GIM			CFF			CG-Nível		
Médias	10%	50%	100%	10%	50%	100%	10%	50%	100%
Nós	34.793	32.381	31.682	20.890	15.929	16.760	33.242	30.302	29.770
Cortes gerados	53.168	78.322	85.169	77.737	258.641	310.867	48.568	76.603	66.217
Cortes inseridos	30.504	29.536	30.230	29.347	28.039	28.704	28.660	27.993	26.792

Tabela 3.4: Variação do tamanho do *round*

também implica na inserção de cortes possivelmente menos eficazes, já que os cortes são ordenados pelo critério de eficiência. Assim, um *round* menor pode já concentrar a maioria dos cortes que são inseridos no PL após a reotimização.

Critérios de Seleção

Este teste busca averiguar a eficiência dos dois possíveis critérios para a seleção de cortes expostos na Seção 1.5. Goycoolea (2006) sugere que a seleção por eficiência produz melhores resultados, mas que por outro lado é computacionalmente mais custosa.

Todos os parâmetros do teste, com exceção do critério de seleção, são os mesmos do teste anterior, e é utilizado o tamanho do *round* 10%, que apresentou melhores resultados.

A Tabela 3.5 resume os resultados obtidos. Pode-se observar, no conjunto de instâncias proposto, que o critério de seleção por eficiência obteve melhores resultados do que o critério de seleção por violação, o que sugere que o seu maior custo computacional é recompensado.

	GIM		CFF		CG-Nível	
Instância	viol.	efic.	viol.	efic.	viol.	efic.
mkp10.100-00	60,44	58,96	49,53	49,65	54,65	54,48
mkp10.100-10	61,53	67,51	56,07	57,06	61,68	60,78
mkp10.250-00	27,98	27,88	22,35	21,02	24,77	24,50
mkp10.250-10	34,03	34,83	23,35	28,41	30,65	31,07
mkp10.250-20	38,45	37,86	29,28	22,34	33,39	31,62
air04	33,08	25,44	9,05	29,63	10,66	26,29
air05	45,54	58,69	35,25	47,67	31,26	48,05
harp2	46,48	53,69	60,47	70,38	55,44	68,53
p2756	3,19	3,07	3,38	2,13	4,52	3,24
b05200	43,88	31,72	22,93	54,62	52,55	100,00
c10100	63,28	66,05	37,63	56,24	54,88	41,62
d05200	19,11	14,38	2,47	2,48	10,19	9,55
Média	**39,75**	**40,01**	**29,31**	**36,80**	**35,39**	**41,64**

Tabela 3.5: % Gap fechado para os critérios de seleção violação normalizada e eficiência

Uso de Descarte por Paralelismo

Este teste verifica a eficácia do descarte por paralelismo. A verificação de paralelismo requer um grande número de operações, pois é feita entre o corte gerado e todos os outros cortes já inseridos. Por outro lado, a inserção de um corte paralelo a outro já existente acarreta um aumento desnecessário do problema. O paralelismo entre cortes é medido segundo (1.23) e, entre cortes semelhantes, é descartado o de menor eficiência.

A Tabela 3.6 mostra os resultados obtidos para os mesmos parâmetros do teste de *round* (10%), com e sem uso de descarte por paralelismo. Nesta tabela, *max_par* $= 1$, ou seja, 100% de paralelismo é aceito e nenhum corte é excluído. Para *max_par* $= 0{,}99$, cortes semelhantes são descartados. Os resultados demonstram que o descarte de cortes por paralelismo é eficaz, aumentando significativamente o gap fechado para a maioria das instâncias.

A Tabela 3.7 mostra que a checagem de paralelismo gera um grande aumento da quantidade de nós avaliados, e ainda assim, diminui a quantidade total de cortes inseridos antes da reotimização.

	GIM		CFF		CG-Nível	
Instância	max_par=1	max_par=0.99	max_par=1	max_par=0.99	max_par=1	max_par=0.99
mkp10.100-00	54,47	58,96	45,36	49,65	46,02	54,48
mkp10.100-10	67,57	67,51	52,87	57,06	53,45	60,78
mkp10.250-00	20,00	27,88	16,85	21,02	13,90	24,50
mkp10.250-10	27,14	34,83	19,83	28,41	18,95	31,07
mkp10.250-20	27,48	37,86	20,95	22,34	23,01	31,62
air04	12,39	25,44	49,37	29,63	32,69	26,29
air05	41,07	58,69	57,55	47,67	50,56	48,05
harp2	68,65	53,69	48,24	70,37	75,99	68,53
p2756	1,77	3,07	3,38	2,13	2,13	3,24
b05200	22,70	31,72	11,33	54,62	18,17	100,00
c10100	60,48	66,05	50,31	56,24	67,76	41,62
d05200	10,74	14,38	2,48	2,48	5,85	9,55
Média	34,54	40,01	31,54	36,80	34,04	41,64

Tabela 3.6: % Gap fechado com e sem uso de descarte por paralelismo

	GIM			CFF			CG-Nível		
Médias	par=1	par=0.99	Var.	par=1	par=0.99	Var.	par=1	par=0.99	Var.
Nós	12816	20890	63%	14400	34793	142%	13368	33242	149%
Cortes gerados	89288	77737	-13%	83872	53168	-37%	80695	48568	-40%
Cortes inseridos	27354	29347	7%	27034	30504	13%	27165	28660	6%

Tabela 3.7: Uso de descarte por paralelismo

3.5.3 Parâmetros Relativos a Cortes

No Capítulo 2 são descritos os parâmetros relativos aos cortes CFF, k-cortes e CG-Nível. Nesta seção propõe-se testes relativos ao uso destes parâmetros utilizando-se o mesmo conjunto de instâncias da seção anterior.

3.5.4 Corte Fracionário Forte

O algoritmo ALG-CFF, apresentado na Seção 2.2.1 é usado para analisar a variação de parâmetros do corte CFF.

O primeiro teste é feito multiplicando-se as linhas por -1 antes da geração do corte, sempre que a parte fracionária do lado direito for $< 1/2$ para obter uma nova linha cujo $k = 1$, ou seja *CFF-Kmax* $= 1$.

No segundo teste, propõe-se a limitação de k tomando a metade do máximo valor possível para k. Como já definido na Seção 3.5.2, *frac_rhs* $= 0{,}05$, e portanto, $1 \leq k \leq \lceil 1/\textit{frac\_rhs} \rceil - 1$, ou seja, $1 \leq k \leq 19$. Desta forma, *CFF-Kmax* = $\lfloor 19/2 \rfloor = 9$. Assim, para todo $k > 9$ é aplicada a pré-multiplicação por -1, de forma que o novo corte é obtido com $k = 1$, e para as demais linhas que apresentarem $k \leq 9$, o corte é aplicado sem modificações.

O terceiro teste consiste em usar o parâmetro *CFF-Kmax* = $+\infty$, ou seja, k não está limitado e não é aplicada nenhuma modificação à linha do tableau antes da geração do corte CFF.

A Tabela 3.8 mostra os resultados obtidos para este teste. Nota-se que o parâmetro *CFF-Kmax* = 1, que equivale à forma como Letchford e Lodi (2002) aplicaram o corte, gera os melhores resultados. Note que o gap médio fechado é menor para a estratégia que limita k parcialmente, de onde conclui-se que a imposição de uma limitação deste parâmetro não é interessante.

Instância	CFF-Kmax=1	CFF-Kmax=9	CFF-Kmax=$+\infty$
mkp10.100-00	49,65	47,3	46,9
mkp10.100-10	57,06	51,68	53,57
mkp10.250-00	21,02	15,57	16,78
mkp10.250-10	28,41	22,4	20,13
mkp10.250-20	22,34	21,69	22,18
air04	29,63	9,05	26,6
air05	47,67	47,79	52,15
harp2	70,38	40,75	55,46
p2756	2,13	1,77	2,303
b05200	54,62	52,3	52,9
c10100	56,24	61,7	62,24
d05200	2,48	5,85	5,86
Média	36,80	31,49	34,76

Tabela 3.8: Variação do máximo k para o corte CFF

3.5.5 K-cortes

O algoritmo ALG-K, apresentado na Seção 2.3, é usado para analisar a variação de parâmetros dos k-cortes. São usados três valores de MAXCL. Este parâmetro determina o máximo de k-cortes gerados por linha do tableau, sendo o melhor corte escolhido para inserção. A Tabela 3.9 mostra os resultados obtidos para este teste. O valor de MAXCL = 3 gerou os melhores resultados, o que indica que é mais interessante testar menos valores de k por linha do tableau.

Instância	MAXCL=3	MAXCL=5	MAXCL=10
mkp10.100-00	71,00	72,31	71,36
mkp10.100-10	86,53	85,08	83,37
mkp10.250-00	30,19	29,61	33,01
mkp10.250-10	37,52	36,60	35,00
mkp10.250-20	39,93	39,90	42,18
air04	18,73	28,05	15,50
air05	40,75	34,38	34,38
harp2	52,08	48,05	53,97
p2756	5,35	7,77	4,39
b05200	100,00	100,00	100,00
c10100	79,97	52,94	64,70
d05200	14,76	15,68	14,24
Média	48,07	45,86	46,01

Tabela 3.9: Variações do máximo de cortes por linha para o k-corte

3.5.6 Corte Chvátal-Gomory-Nível

A geração do corte CG-Nível implica na determinação dos parâmetros p e d. O algoritmo ALG-CGN define, na Seção 2.4.2, o número de conjuntos de parâmetros (p, d) utilizados na geração de cortes para uma mesma linha do tableau. Para tanto, recebe como parâmetro, além da linha do tableau, os valores de p_{min} , p_{max} e MAXPY. O valor de MAXPY determina o máximo de valores atribuídos a y, dado um parâmetro p. Para testar o impacto da variação destes parâmetros foram utilizados os valores da Tabela 3.10, e $p_{min} = 1$.

(p_{max}, MAXPY)	Δp
(3, 1)	3
(3, 3)	1
(8, 1)	8
(8, 4)	2
(8, 8)	1

Tabela 3.10: Parâmetros de teste para ALG-CGN

A Tabela 3.11 apresenta o resultado da variação destes parâmetros para o conjunto de 12 instâncias utilizados nos testes anteriores, utilizando o algoritmo de duas fases. É esperado que uma faixa mais ampla de parâmetros permita a geração de cortes mais fortes, mas que também aumente o tempo gasto na geração de cortes. Neste teste, o conjunto de parâmetros [8, 4] apresentou o melhor resultado.

Instância	[3,1]	[8,1]	[3,3]	[8,8]	[8,4]
mkp10.100-00	56,09	56,06	54,41	51,05	53,88
mkp10.100-10	65,65	65,62	60,78	62,23	64,66
mkp10.250-00	23,21	23,16	24,23	22,60	24,71
mkp10.250-10	29,62	29,58	31,04	29,06	30,51
mkp10.250-20	33,16	33,1	31,61	31,3	32,33
air04	29,71	29,7	26,3	29,7	30,4
air05	45,77	45,77	48,05	43,71	39,52
harp2	70,98	70,98	68,73	61,74	71,97
p2756	2,77	2,44	4	1,81	1,82
b05200	100	100	100	100	100
c10100	39,49	39,94	42,43	29,62	57,13
d05200	12,41	12,39	9,74	9,52	9,57
Média	42,41	42,40	41,78	39,36	43,04

Tabela 3.11: Resultados de teste para ALG-CGN

O próximo teste tem a função de medir o ganho proporcionado pela aplicação dos algoritmos de duas fases do corte CG-Nível. A Tabela 3.12 apresenta o gap de integralidade fechado para o mesmo conjunto de instâncias e parâmetros do teste de *round* (10%) que utiliza segunda fase.

Note que a aplicação dos algoritmos de duas fases não provocaram grande diferença no gap fechado para estas instâncias. Isto pode ser explicado pelo fato de que o ganho proporcionado pela segunda fase não é compensado pelo tempo gasto para recalcular os parâmetros do corte. O fato de os problemas testados possuírem muitas variáveis torna mais difícil encontrar um parâmetro que atenda às condições de todos os coeficientes de um dado sinal, e ao mesmo tempo possibilite a modificação de uma parte dos coeficientes do sinal oposto.

3.6 Testes Comparativos

Esta seção apresenta a comparação entre os planos de corte citados para um conjunto de 40 instâncias. A partir dos resultados das seções anteriores, o conjunto de parâmetros usados é mostrado na Tabela 3.13. O tempo máximo de execução é de 300 segundos. O pré-processamento do CPLEX é habilitado, assim como a opção *strong branching*.

A inserção dos cortes GIM e k-cortes exige a adição de uma nova variável de folga real. Para aplicar estes cortes utilizando-se apenas variáveis inteiras, propõe-se a substituição das variáveis de

Instância	2 fases	1 fase
mkp10.100-00	54,48	54,38
mkp10.100-10	60,78	60,72
mkp10.250-00	24,50	24,22
mkp10.250-10	31,06	31,15
mkp10.250-20	31,62	31,67
air04	26,29	23,29
air05	48,05	48,05
harp2	68,53	68,53
p2756	3,24	3,4
b05200	100	100
c10100	41,62	42,44
d05200	9,55	9,48
Média	41,64	41,44

Tabela 3.12: Comparação entre o uso de 1 ou 2 fases ($p_{max} = 3$, MAXPY $= 3$)

folga reais antes da derivação do corte. Desta forma, tem-se os cortes GIF e k-cortes (GIF), respectivamente.

Parâmetro	Valor
frac_rhs	0,05
max_round_percent	10%
viol_min	10^{-4}
max_razao	10^5
max_nao_nulos	100%
max_par	99%
rhs_relax	0
crit_sel	eficiência
CFF	
CFF-Kmax	1
k-corte	
MAXCL	3
CG-Nivel	
p_{max}	8
MAXPY	4
duas fases	sim

Tabela 3.13: Parâmetros utilizados nos testes da Seção 3.6

As relações de dominância existentes entre os cortes são do corte GIF, k-cortes (GIF) e CFF em

relação ao corte fracionário (GI). Portanto é esperado que estes cortes superem o corte GI em relação ao gap de integralidade fechado.

Note que não há como comparar o corte GIM com os cortes para programação inteira pura, pois o corte GIM é aplicado sobre variáveis reais. Ainda assim, quando este corte (GIM) é aplicado apenas sobre variáveis inteiras, ele é equivalente a GIF e, portanto, domina GI. O mesmo vale para os k-cortes. Portanto, é esperado que o gap fechado por GIM e k-cortes com variáveis reais também seja superior ao gap fechado pelos cortes fracionários.

A Tabela 3.14 mostra o gap de integralidade fechado pela aplicação dos planos de corte a cada instância, enquanto a Tabela 3.15 mostra o gap de integralidade fechado para cada grupo de instância e a Tabela 3.16 mostra a quantidade de nós avaliados e cortes inseridos. A Tabela 3.17 mostra a relação de cortes por nó.

Como esperado, a Tabela 3.14 mostra que os cortes GIM, k-cortes e CFF possibilitam um maior fechamento do gap que o corte fracionário (GI). Dentre estes três cortes, o k-corte possibilitou o maior gap médio fechado para este conjunto de instâncias. Os cortes GIF e k-cortes (GIF) apresentam gap médio fechado maior do que os cortes equivalentes para programação inteira mista.

O corte CG-Nível supera os cortes CFF e os cortes fracionários no fechamento do gap de integralidade. Note que não há relação de dominância entre o corte CG-Nível e os demais. Mas o corte CG-Nível possibilita o uso de variação de parâmetros através da aplicação do algoritmo ALG-CGN, de modo a selecionar cortes mais fortes a partir da mesma linha. Os cortes CFF também apresentam gap médio fechado maior que os cortes fracionários, o que é esperado por serem teoricamente superiores.

O comportamento dos planos de corte em relação ao gap fechado é semelhante para as instâncias da mochila multidimensional, da mochila multidimensional com restrições de demanda e da MIPLIB. Os cortes GIF e k-cortes (GIF) superam os cortes mistos correspondentes, GIM e k-cortes, no fechamento do gap médio de integralidade. Os demais cortes apresentam resultados inferiores aos cortes mistos. Esta mesma relação pode ser observada para instâncias de designação generalizada, mas a diferença entre os cortes GIF e k-cortes (GIF) e os demais é mais acentuada.

Observe ainda que a razão de cortes por nó exibida na Tabela 3.17 é menor para os cortes GIF e k-cortes (GIF), para todos os tipos de instância. Estes cortes são seguidos pelos cortes mistos, GIM e k-cortes. O fato de que os cortes cuja razão de cortes por nó é menor apresentam, em geral, maior gap médio fechado indica que, para os demais cortes, o problema linear de cada nó pode ter se tornado de difícil solução, e o benefício proveniente da inserção de cortes pode ter sido diminuído pelo tempo gasto na resolução do PL.

Observe na Tabela 3.16 que para problemas da mochila multidimensional, a quantidade média de cortes é semelhante. Entretanto, a média de nós avaliados é menor para os cortes cujo fechamento do gap é menor (Tabela 3.15). Este fato é refletido na razão de cortes por nó da Tabela 3.17, indicando que a inserção de cortes sem grande ganho no fechamento do gap de integralidade leva a um resultado final inferior.

3.7 Conclusões

A partir dos testes apresentados na Seção 3.5.1, fica claro que o uso da técnica IC gera melhores resultados do que a aplicação da técnica IR. Esta técnica possibilita que cortes cuja variável de folga

é básica, e que, portanto, não estão ativos, não sejam inseridos no PL da árvore de *branch-and-cut*. Desta forma, o problema linear resultante torna-se mais fácil de ser resolvido, e mais nós são avaliados em um mesmo intervalo de tempo. O percentual de gap fechado ao final é sempre maior para IR, indicando que a inserção de cortes acarretada pela técnica IC é excessiva.

Ainda observando os resultados da comparação entre IC e IR, conclui-se que os valores adotados na literatura para os parâmetros relacionados a erros de precisão, *max_razao* e *viol_min* são adequados, pois estes mesmos valores foram usados no teste indicado sem gerar erros de precisão numérica. Além disso, o parâmetro que procura adicionar folga ao corte, *rhs_relax*, mostrou-se desnecessário para todos os cortes, inclusive aqueles com variável de folga real.

Os testes também demonstram que o uso do critério de descarte por paralelismo aumenta o percentual de gap fechado para todos os planos de corte, ao evitar a adição de cortes redundantes.

Entre os dois critérios de seleção utilizados, o critério de eficiência mostrou os melhores resultados. Apesar de apresentar maior custo computacional do que o critério de violação normalizada, o percentual de gap fechado ao final da execução é maior para este critério.

Balas et al. (1996b) mostram resultados relativos ao tamanho do *round* de cortes, utilizando os mesmos valores deste trabalho: 10%, 50% e 100%. Os autores concluem que o *round* completo gera melhores resultados. Entretanto, os testes feitos pelos autores referem-se apenas ao nó raiz. Neste trabalho, os resultados apontam para o uso do tamanho de *round* 10%.

Testes de parâmetros foram executados para o corte fracionário forte (CFF), K-cortes e para o corte CG-Nível. O corte CFF apresentou melhores resultados quando implementado segundo propuseram os autores em (Letchford e Lodi, 2002). A pré-multiplicação por -1 antes da geração do corte é um procedimento heurístico que mostra resultados satisfatórios experimentalmente. Para os K-cortes, a geração de no máximo três cortes por linha apresentou melhores resultados em relação ao gap fechado.

Para o corte CG-Nível, o teste relativo aos parâmetros p e d mostraram que uma faixa mais ampla de parâmetros permite a geração de cortes mais fortes, proporcionando um maior fechamento do gap médio. A aplicação das estratégias de duas fases mostrou uma pequena melhoria no gap médio final, indicando pequeno ganho proporcionado pela segunda fase ao se reajustar coeficientes de um dado sinal.

Os teste comparativos realizados mostram um maior fechamento do gap médio para os cortes GIM, GIF, K-cortes e K-cortes (GIF). Os cortes calculados apenas sobre variáveis inteiras, GIF e K-cortes (GIF), mostram o maior gap médio fechado. Estes cortes superaram o corte fracionário GI. Este resultado já era esperado, pois os cortes GIF e K-cortes (GIF) dominam o corte GI. O mesmo resultado também se confirmou a partir da aplicação dos cortes análogos, com variáveis reais, GIM e K-cortes, que dominam os cortes fracionários quando aplicados apenas sobre variáveis inteiras.

Os cortes CFF dominam os cortes fracionários, e o gap médio fechado também é maior para este corte. Já o corte CG-Nível não apresenta relação de dominância em relação aos demais cortes, mas o seu desempenho é superior ao dos corte fracionário e do corte CFF. Tal comportamento pode ser explicado por ser um corte mais maleável em relação à variação de parâmetros.

Os resultados apontam que a introdução de qualquer corte deve ser criteriosa. O tempo gasto na resolução do PL de cada nó pode superar o ganho proporcionado pela inserção de novos cortes.

	Gap fechado(%)						
Instância	GI	GIM	GIF	CFF	K-corte	K-corte(GIF)	CG-Nível
5-250-1-0-1	14,95	34,22	35,09	12,87	31,56	33,01	22,69
5-250-1-0-3	15,66	38,50	35,89	15,11	28,80	29,82	16,28
5-250-1-0-11	12,98	31,27	34,04	12,15	34,01	28,94	14,18
5-250-1-1-0	21,46	36,63	41,37	20,97	42,12	40,55	20,94
5-250-1-1-2	14,02	31,98	30,47	15,44	33,32	28,65	14,07
5-250-1-1-6	20,25	35,75	35,77	19,94	30,09	34,57	17,90
10-100-1-0-0	36,83	41,14	47,53	29,46	41,52	49,28	38,88
10-100-1-0-1	31,91	40,48	47,06	32,25	39,08	44,66	33,01
10-100-1-1-14	52,63	61,73	70,05	56,69	67,45	82,60	60,22
10-100-5-0-10	31,29	38,20	46,59	33,00	38,37	47,01	30,73
10-100-5-1-14	34,45	49,80	51,49	37,88	49,72	50,84	32,43
10-100-10-0-0	41,23	51,09	54,61	86,93	49,19	69,81	39,30
cap6000	63,55	63,55	63,55	63,55	67,05	63,55	63,55
fast0507	7,59	7,59	7,59	7,59	7,59	7,59	7,59
harp2	22,69	60,98	75,16	40,46	53,30	76,39	43,50
manna81	22,93	22,93	22,93	22,93	22,93	22,93	24,06
p2756	100,00	100,00	100,00	100,00	100,00	100,00	95,35
mkp5.500-06	14,84	22,83	22,48	16,57	22,69	21,56	24,58
mkp5.500-16	13,34	28,31	25,84	10,74	34,38	19,54	25,14
mkp5.500-26	22,83	36,30	40,70	24,02	43,04	31,41	34,51
mkp10.100-07	74,84	100,00	100,00	76,00	100,00	100,00	87,86
mkp10.100-17	67,49	79,76	100,00	67,22	81,61	100,00	72,77
mkp10.250-05	16,45	23,93	27,80	16,30	24,33	27,42	18,39
mkp10.250-15	23,21	27,63	32,60	20,84	27,81	32,05	24,75
mkp10.250-25	16,95	37,48	41,79	26,45	37,45	41,02	31,06
mkp10.500-00	1,70	10,02	11,29	4,23	10,73	10,91	6,25
mkp10.500-11	4,62	12,07	13,47	3,00	12,18	13,08	8,75
mkp10.500-21	3,82	12,75	13,44	3,06	12,92	13,17	7,54
b10200	3,22	3,22	38,70	3,22	3,22	36,94	20,07
b20100	100,00	100,00	100,00	34,45	100,00	100,00	58,57
b20200	2,42	52,70	81,51	2,42	38,58	100,00	2,42
c05200	15,39	2,24	100,00	2,24	31,90	100,00	28,51
c10200	4,32	17,40	38,43	4,32	4,32	37,52	4,32
c20200	2,75	2,75	32,82	2,75	2,75	34,96	2,75
d05100	14,42	24,52	41,70	13,40	20,07	43,99	19,22
d10200	1,62	1,62	11,12	1,62	1,62	11,07	1,62
e05100	33,97	67,19	100,00	31,29	67,99	100,00	47,43
e05200	22,10	32,03	100,00	32,06	40,01	100,00	22,10
e10100	3,75	13,29	39,15	31,08	12,82	42,50	3,75
e10200	5,99	5,08	25,86	5,99	5,08	27,61	5,99
Média	25,36	36,47	48,45	26,01	36,79	48,87	28,33

Tabela 3.14: % Gap fechado

	Média de gap (%)			
Corte	MDMKP	MKP	MIPLIB	GAP
Gomory	27,31	23,64	43,35	17,49
GIM	40,90	35,55	51,01	26,84
GIF	44,16	39,04	53,85	59,11
CFF	31,06	24,40	46,91	13,74
K-corte	40,44	37,01	50,17	27,36
K-corte (GIF)	44,98	37,29	54,09	61,22
CG-Nível	28,39	31,05	46,81	18,06

Tabela 3.15: Média do gap fechado por tipo de instância

	Média de nós				Média de cortes			
Corte	MDMKP	MKP	MIPLIB	GAP	MDMKP	MKP	MIPLIB	GAP
Gomory	5581	6446	182	393	11563	8195	392	869
GIM	14018	16978	192	1092	14834	8874	388	1245
GIF	35847	25704	1453	8132	13982	7768	818	1676
CFF	5654	6266	229	419	11463	8674	367	837
K-corte	13298	17653	183	1088	14456	9488	372	1236
K-corte (GIF)	31154	22345	1202	8124	13457	7634	686	1916
CG-Nível	5561	13140	201	567	8662	8204	525	757

Tabela 3.16: Médias para cada plano de corte

Corte	MDMKP	MKP	MIPLIB	GAP
Gomory	2,07	1,27	2,16	2,21
GIM	1,06	0,52	2,02	1,14
GIF	0,39	0,30	0,56	0,21
CFF	2,03	1,38	1,60	2,00
K-corte	1,09	0,54	2,04	1,14
K-corte (GIF)	0,43	0,34	0,57	0,24
CG-Nível	1,56	0,62	2,61	1,34

Tabela 3.17: Razão de cortes por nó

Referências Bibliográficas

Achterberg, T., Koch, T., e Martin, A. (2003). MIPLIB 2003. *http://miplib.zib.de*.

Achterberg, T., Koch, T., e Martin, A. (2006). MIPLIB 2003. *Operations research letters*, 34(4):361–372.

Andersen, K., Cornuéjols, G., e Li, Y. (2005). Reduce-and-Split Cuts: Improving the Performance of Mixed-Integer Gomory Cuts. *Management Science*, 51(11):1720–1732.

Andreello, G., Caprara, A., e Fischetti, M. (2007). Embedding $\{0, 1/2\}$-Cuts in a Branch-and-Cut Framework: A Computational Study. *INFORMS Journal on Computing*, 19(2):229–238.

Arntzen, H., Hvattum, L., e Løkketangen, A. (2006). Adaptive memory search for multidemand multidimensional knapsack problems. *Computers and Operations Research*, 33(9):2508–2525.

Balas, E. (1979). Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51.

Balas, E., Ceria, S., e Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58(1):295–324.

Balas, E., Ceria, S., e Cornuéjols, G. (1996a). Mixed 0-1 Programming by Lift-and-Project in a Branch–and–Cut Framework. *Management Science*, 42(9):1229–1246.

Balas, E., Ceria, S., Cornuéjols, G., e Natraj, N. (1996b). Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9.

Balas, E. e Jeroslow, R. G. (1980). Strengthening cuts for mixed integer programs. *European Journal of Operational Research*, 4(4):224–234.

Beasley, J. E. (1990). OR-Library. *http://people.brunel.ac.uk/~mastjjb/jeb/info.html*.

Ben-Israel, A. e Charnes, A. (1962). On some problems of diophantine programming. *Cahiers du Centre d'Etudes de Recherche Opérationnelle*, 4:215–280.

Bixby, R. E., Ceria, S., McZeal, C. M., e Savelsbergh, M. W. P. (1996). MIPLIB 3.0. *http://www.caam.rice.edu/~bixby/miplib/miplib.html*.

Caprara, A. e Fischetti, M. (1996). $\{O, 1/2\}$-Chvátal-Gomory cuts. *Mathematical programming*, 74(3):221–235.

Caprara, A., Fischetti, M., e Letchford, A. N. (2000). On the separation of maximally violated mod-k cuts. *Mathematical Programming*, 87(1):37–56.

Ceria, S., Cornuéjols, G., e Dawande, M. (1995). Combining and Strengthening Gomory Cuts. *Lecture Notes in Computer Science*, 920:438–451.

Chu, P. C. e Beasley, J. E. (1998). A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, 4(1):63–86.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., e Schrijver, A. (1998). *Combinatorial optimization*. Wiley, New York.

Cornuéjols, G. (2007). Revival of the Gomory cuts in the 1990's. *Annals of Operations Research*, 149(1):63–66.

Cornuéjols, G. (2008). Valid Inequalities for Mixed Integer Linear Programs. *Mathematical Programming*, 112(1):3–44.

Cornuéjols, G. e Li, Y. (2001). Elementary closures for integer programs. *Operations Research Letters*, 28(1):1–8.

Cornuéjols, G., Li, Y., e Vandenbussche, D. (2003). K-Cuts: A Variation of Gomory Mixed Integer Cuts from the LP Tableau. *INFORMS Journal on Computing*, 15(4):385–396.

Eisenbrand, F. (1999). On the Membership Problem for the Elementary Closure of a Polyhedron. *Combinatorica*, 19(2):297–300.

Garfinkel, R. S. e Nemhauser, G. L. (1972). *Integer Programming*. Wiley, New York.

Glover, F. (1966). Generalized Cuts in Diophantine Programming. *Management Science*, 13(3):254–268.

Glover, F. e Sherali, H. D. (2005). Chvatal–Gomory–tier cuts for general integer programs. *Discrete Optimization*, 2(1):51–69.

Gomory, R. E. (1958). Outline of an Algorithm for Integer Solution to Linear Programs. *Bulletin of the American Mathematical Society*, 64(5):275–278.

Gomory, R. E. (1960a). An algorithm for the mixed-integer problem. *RM-2597, Rand Corporation*.

Gomory, R. E. (1960b). Solving Linear Programming Problems in Integers. Em Bellman, R. E. e Jr., M. H., eds., *Combinatorial Analysis*, páginas 211–216. American Mathematical Society.

Gomory, R. E. (1963a). An Algorithm for Integer Solutions to Linear Programs. Em Graves, R. e Wolfe, P., eds., *Recent Advances in Mathematical Programming*, páginas 269–302. McGraw-Hill.

Gomory, R. E. (1963b). An All-Integer Programming Algorithm. Em Muth, J. F. e Thompson, G. I., eds., *Industrial Scheduling*, páginas 193–206. Prentice-Hall.

Gomory, R. E. (1965). On the Relation between Integer and Noninteger Solutions to Linear Programs. *Proceedings of the National Academy of Sciences*, 53(2):260–265.

Gomory, R. E. (1967). Faces of an Integer Polyhedron. *Proceedings of the National Academy of Sciences*, 57(1):16–18.

Gomory, R. E. (1969). Some Polyhedra Related to Combinatorial Problems. *Linear Algebra and its Applications*, 2(4):451–558.

Gomory, R. E. (1970). Properties of a Class of Integer Polyhedra. Em Abadie, J., editor, *Integer and Nonlinear Programming*, páginas 353–365. North-Holland.

Gomory, R. E. (2002). Early Integer Programming. *Operations Research*, 50(1):78–81.

Gomory, R. E. e Hoffman, A. (1963). On the convergence of an integer programming process. *Naval Research Logistics Quarterly*, 10:121–124.

Gomory, R. E. e Johnson, E. L. (1972). Some continuous functions related to corner polyhedra, II. *Mathematical Programming*, 3(1):359–389.

Gomory, R. E., Johnson, E. L., e Evans, L. (2003). Corner Polyhedra and their connection with cutting planes. *Mathematical Programming*, 96(2):321–339.

Goycoolea, M. G. (2006). Cutting Planes for Large Mixed Integer Programming Models. Tese de doutorado. *School of Industrial and Systems Engineering, Georgia Institute of Technology*.

Günlük, O. e Pochet, Y. (2001). Mixing mixed-integer inequalities. *Mathematical Programming*, 90(3):429–457.

Hartmann, M., Queyranne, M., e Wang, Y. (1999). On the Chvátal rank of certain inequalities. Em Cornuéjols, G., Burkard, R. E., e Woeginger, G. J., eds., *Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science*, volume 1610, páginas 218–233. Springer.

ILOG (2006a). ILOG CPLEX 10.1: C API Reference Manual.

ILOG (2006b). ILOG CPLEX 10.1: ILOG CPLEX 10.1 Parameters.

Letchford, A. N. (2002). Totally Tight Chvatal–Gomory Cuts. *Operations Research Letters*, 30(2):71–73.

Letchford, A. N. e Lodi, A. (2002). Strengthening Chvátal-Gomory cuts and Gomory fractional cuts. *Operations Research Letters*, 30(2):74–82.

Lovász, L. e Schrijver, A. (1991). Cones of Matrices and Set-Functions, and 0-1 Optimization. *SIAM Journal on Optimization*, 1:166–190.

Marchand, H. e Wolsey, L. A. (2001). Aggregation and Mixed Integer Rounding to Solve MIPs. *Operations Research*, 49(3):363–371.

Martin, G. (1963). An accelerated euclidean algorithm for integer linear programming. Em Graves, R. L. e Wolfe, P., eds., *Recent Advances in Mathematical Programming*, páginas 311–318, New York. McGraw–Hill.

Miliotis, P. (1978). Using cutting planes to solve the symmetric traveling salesman problem. *Mathematical Programming*, 15(1):177–188.

Nemhauser, G. L. e Wolsey, L. A. (1988). *Integer and combinatorial optimization*. Wiley, New York.

Nemhauser, G. L. e Wolsey, L. A. (1990). A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Mathematical Programming*, 46(1):379–390.

Padberg, M. e Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large–Scale Symmetric Traveling Salesman Problems. *SIAM Review*, 33(1):60–100.

Wolsey, L. A. (1998). *Integer Programming*. Wiley, New York.

Yagiura, M., Ibaraki, T., e Glover, F. (2006). A path relinking approach with ejection chains for the generalized assignment problem. *European journal of operational research*, 169(2):548–569.

Apêndice A

Complementação do Capítulo 2

Este apêndice detalha algumas passagens do Capítulo 2.

A.1 Proposição 2.3

A prova desta proposição é facilitada pela definição dos conceitos e lemas colocados a seguir.

Seja a^k o termo a_i^k na equação recursiva (2.24), suprimido o índice i. Nos lemas a seguir, considera-se que

$$\beta^k \equiv \left\lceil \frac{a^k}{d^k} \right\rceil d^k - a^k \text{ , e} \tag{A.1}$$

$$\gamma^k \equiv \beta^k - \left\lfloor \frac{\beta^k}{d^{k+1}} \right\rfloor d^{k+1} . \tag{A.2}$$

Ambas as definições acima são sempre positivas. Dividindo-se a expressão de β^k por d^k, obtém-se

$$\beta^k/d^k = \left\lceil a^k/d^k \right\rceil - a^k/d^k.$$

Observe que β^k/d^k é a diferença entre o teto de um número e ele mesmo, e, portanto, $\beta^k/d^k \geq 0$. Como $d^k > 0$, então $\beta^k \geq 0$. Analogamente, sabendo que $d^{k+1} > 0$, obtém-se

$$\gamma^k/d^{k+1} = \beta^k/d^{k+1} - \left\lfloor \beta^k/d^{k+1} \right\rfloor ,$$

e, portanto, $\gamma^k \geq 0$.

Também é possível mostrar que $d^k > \beta^k$ e $d^{k+1} > \gamma^k$. Observe que $d^k > \beta^k$ é equivalente a

$$d^k > \left\lceil a^k/d^k \right\rceil /d^k - a^k.$$

Dividindo-se esta equação por d^k, obtém-se

$$1 > \left\lceil a^k/d^k \right\rceil - a^k/d^k,$$

e, portanto, $d^k > \beta^k$. De maneira análoga, $d^{k+1} > \gamma^k$ implica que

$$d^{k+1} > \beta^k - \left\lfloor \beta^k/d^{k+1} \right\rfloor d^{k+1}, \text{ ou}$$

$$1 > \beta^k/d^{k+1} - \left\lfloor \beta^k/d^{k+1} \right\rfloor,$$

e, portanto, $d^{k+1} > \gamma^k$.

Em suma, β^k e γ^k são definidos de forma que

$$d^k > \beta^k \geq 0 \tag{A.3}$$

e

$$d^{k+1} > \gamma^k \geq 0. \tag{A.4}$$

Lema A.1. $\left\lceil a^{k+1}/d^{k+1} \right\rceil = \left\lceil a^k/d^k \right\rceil - \left\lfloor \beta^k/d^{k+1} \right\rfloor$.

Demonstração: Dividindo a equação (2.24) por d^{k+1}, obtemos

$$a^{k+1}/d^{k+1} = a^k/d^{k+1} - \left\lceil a^k/d^k \right\rceil \left(1/d^{k+1}\right).$$

A partir da definição de β^k (equação A.1), tem-se $a^k = \left\lceil a^k/d^k \right\rceil d^k - \beta^k$ que, se divido por d^{k+1} resulta em

$$a^k/d^{k+1} = \left\lceil a^k/d^k \right\rceil \left(d^k/d^{k+1}\right) - \beta^k/d^{k+1}.$$

Substituindo este termo na expressão de a^{k+1}/d^{k+1}, temos:

$$\begin{aligned} a^{k+1}/d^{k+1} &= \left\lceil a^k/d^k \right\rceil \left(d^k/d^{k+1}\right) - \beta^k/d^{k+1} - \left\lceil a^k/d^k \right\rceil \left(1/d^{k+1}\right) \\ &= \left\lceil a^k/d^k \right\rceil \left[\left(d^k - 1\right)/d^{k+1}\right] - \beta^k/d^{k+1}. \end{aligned}$$

Mas, pela equação (2.23), $d^{k+1} = d^k - 1$ e, então, a equação acima torna-se

$$\begin{aligned} &a^{k+1}/d^{k+1} = \left\lceil a^k/d^k \right\rceil - \beta^k/d^{k+1}, \text{ ou} \\ &a^{k+1}/d^{k+1} + \beta^k/d^{k+1} = \left\lceil a^k/d^k \right\rceil. \end{aligned}$$

Na equação acima, um número inteiro $\left(\left\lceil a^k/d^k \right\rceil\right)$ é igual à soma de outros dois fracionários. Isto acontece se e somente se a parte fracionária das parcelas $\left(a^{k+1}/d^{k+1}\right.$ e $\left.\beta^k/d^{k+1}\right)$ são complementares, ou seja, têm soma unitária. Sendo assim, podemos escrever que

$$\left\lceil a^{k+1}/d^{k+1} \right\rceil + \left\lfloor \beta^k/d^{k+1} \right\rfloor = \left\lceil a^k/d^k \right\rceil,$$

ou, escrevendo de outra forma,

$$\left\lceil a^{k+1}/d^{k+1} \right\rceil = \left\lceil a^k/d^k \right\rceil - \left\lfloor \beta^k/d^{k+1} \right\rfloor. \quad \square$$

Lema A.2. $\gamma^k = \beta^{k+1}$.

Demonstração: Pela definição de β^k (equação A.1), sabe-se que $\beta^{k+1} = \left\lceil a^{k+1}/d^{k+1} \right\rceil d^{k+1} - a^{k+1}$. Utilizando o resultado do Lema A.1, podemos substituir $\left\lceil a^{k+1}/d^{k+1} \right\rceil$ nesta equação, gerando

$$\beta^{k+1} = \left(\left\lceil a^k/d^k \right\rceil - \left\lfloor \beta^k/d^{k+1} \right\rfloor\right) d^{k+1} - a^{k+1}$$

$$= \lceil a^k/d^k \rceil \, d^{k+1} - \lfloor \beta^k/d^{k+1} \rfloor \, d^{k+1} - a^{k+1}.$$

Como $d^{k+1} = d^k - 1$ e $a^{k+1} = a_k - \lceil a^k/d^k \rceil$, temos

$$\begin{aligned}
\beta^{k+1} &= \lceil a^k/d^k \rceil \, (d^k - 1) - \lfloor \beta^k/d^{k+1} \rfloor \, d^{k+1} - \left(a^k - \lceil a^k/d^k \rceil\right) \\
&= \lceil a^k/d^k \rceil \, d^k - \lceil a^k/d^k \rceil - \lfloor \beta^k/d^{k+1} \rfloor \, d^{k+1} - a^k + \lceil a^k/d^k \rceil \\
&= \left(\lceil a^k/d^k \rceil \, d^k - a^k\right) - \lfloor \beta^k/d^{k+1} \rfloor \, d^{k+1} + \left(\lceil a^k/d^k \rceil - \lceil a^k/d^k \rceil\right) \\
&= \beta^k - \lfloor \beta^k/d^{k+1} \rfloor \, d^{k+1} = \gamma^k. \quad \square
\end{aligned}$$

Lema A.3. Se $d^{k+1} > \beta^k$, então $\lceil a^{k+1}/d^{k+1} \rceil = \lceil a^k/d^k \rceil$ e $\beta^{k+1} = \beta^k$.

Demonstração: Se $d^{k+1} > \beta^k$, então $\lfloor \beta^k/d^{k+1} \rfloor = 0$. Neste caso, pela definição de γ^k (equação A.2), $\gamma^k = \beta^k$. Mas, pelo lema (A.2), $\gamma^k = \beta^{k+1}$. Portanto, $\beta^{k+1} = \beta^k$.

A segunda parte deste lema deriva diretamente do Lema A.1, que diz que $\lceil a^{k+1}/d^{k+1} \rceil = \lceil a^k/d^k \rceil - \lfloor \beta^k/d^{k+1} \rfloor$. Dado que $\lfloor \beta^k/d^{k+1} \rfloor = 0$, conclui-se que $\lceil a^{k+1}/d^{k+1} \rceil = \lceil a^k/d^k \rceil$. $\square$

Lema A.4. . Seja $r = \lceil d - \beta \rceil$, onde $\beta \equiv \beta^0$. Então $\beta^k = \beta$ e $\lceil a^k/d^k \rceil = \lceil a/d \rceil$ para $k \leq r - 1$, e $a^k = a - k \lceil a/d \rceil$ para $k \leq r$.

Demonstração: Note que $r = \lceil d - \beta \rceil = \lfloor d - \beta \rfloor + 1 < d - \beta + 1$. Portanto, $d - (r - 1) > \beta$, ou, $d^{r-1} > \beta$. Como d^k é função decrescente de k, podemos concluir que

$$d^k > \beta, \text{para k } = 0, 1, ..., r - 1.$$

Pelo Lema A.3, $d^{k+1} > \beta^k$ implica $\beta^{k+1} = \beta^k$. Aplica-se recursivamente este lema. Inicialmente,

$d^1 > \beta$, que implica $\beta^1 = \beta$. Portanto,
$d^2 > \beta^1$, que implica $\beta^2 = \beta^1 = \beta$,
...
$d^{r-1} > \beta^{r-2}$, que implica $\beta^{r-1} = \beta$.

E, portanto, podemos concluir que

$$\beta^k = \beta, \text{para k } = 0, 1, ..., r - 1.$$

Também pelo Lema A.3 podemos concluir de forma análoga que

$$\lceil a^k/d^k \rceil = \lceil a/d \rceil \text{ , para k } = 0, 1, ..., r - 1.$$

Observando a equação (2.24), que define a^k recursivamente, podemos escrever que

$$\begin{aligned}
a^1 &= a - \lceil a/d \rceil, \\
a^2 &= a^1 - \lceil a^1/d^1 \rceil = a - 2 \lceil a/d \rceil, \\
&\ldots \\
a^{r-1} &= a - (r - 2) \lceil a/d \rceil - \lceil a/d \rceil = a - (r - 1) \lceil a/d \rceil,
\end{aligned}$$

$$a^r = a^{r-1} - \left\lceil a^{r-1}/d^{r-1} \right\rceil = a - r\left\lceil a/d \right\rceil .$$

A generalização deste procedimento dá origem ao último resultado desta proposição,

$$a^k = a - k\left\lceil a^k/d^k \right\rceil, \text{para k } = 0, 1, ..., r. \quad \square$$

O último lema define o termo r, muito importante na definição do corte CG-Nível. O termo r é uma medida da diferença entre d e β, e portanto, é função do coeficiente a^0 da equação original e do parâmetro d. Este lema consegue definir uma expressão fechada para o coeficiente a^p, desde que $p \leq r$. Os próximos lemas têm a função de encontrar a expressão fechada para a^p quando $p > r$, completando a base para a prova da Proposição 2.3.

Lema A.5. Se $0 < d^{k+1} \leq \beta^k$, então $\beta^{k+1} < 1$.

Demonstração: Pela definição de γ^k na equação (A.2), podemos escrever que

$$\left(\beta^k - \gamma^k\right)/d^{k+1} = \left\lfloor \beta^k/d^{k+1} \right\rfloor .$$

Mas, pela premissa do lema, $0 < d^{k+1} \leq \beta^k$, e portanto, $\left\lfloor \beta^k/d^{k+1} \right\rfloor \geq 1$, originando

$$\left(\beta^k - \gamma^k\right)/d^{k+1} \geq 1,$$

que pode ser reescrita como

$$\gamma^k \leq \beta^k - d^{k+1}.$$

Considerando que $d^{k+1} = d^k - 1$, temos

$$\gamma^k \leq \beta^k - (d^k - 1) = 1 - (d^k - \beta^k).$$

Pela equação (A.3), $d^k > \beta^k$. Portanto, $\gamma^k < 1$. Como o lema A.2 diz que $\gamma^k = \beta^{k+1}$, temos $\beta^{k+1} < 1$. $\square$

Lema A.6. Seja r o mesmo definido no Lema A.4. Então, $\left\lceil a^k/d^k \right\rceil = \left\lceil a^r/d^r \right\rceil$ para $r \leq k \leq p-1$.

Demonstração: Por definição $r = \lceil d - \beta \rceil$. Portanto, $d - r \leq \beta$. Mas, pelo Lema A.4, $\beta^{r-1} = \beta$. Então podemos concluir que $d - r \leq \beta^{r-1}$, ou, pela equação (2.23), $d^r \leq \beta^{r-1}$. Aplicando o Lema A.5, conclui-se que $\beta^r < 1$.

Mas, por definição na equação (2.25), $d^k \geq 1$, para $k \leq p-1$. Então $d^k > \beta^r$, para $k \leq p-1$.

Assim como foi feito no Lema A.4, utilizamos o lema A.3 para analisar que

$d^{r+1} > \beta^r$, que implica $\beta^{r+1} = \beta^r$. Portanto,
$d^{r+2} > \beta^{r+1}$, que implica $\beta^{r+2} = \beta^{r+1} = \beta^r$,
...
$d^{p-1} > \beta^{p-2}$, que implica $\beta^{p-1} = \beta^r$.

E, portanto, podemos concluir que

$$\beta^k = \beta^r, \text{para k } = r, r+1, ..., p-1.$$

Também pelo Lema A.3 podemos concluir de forma análoga que

$$\left\lceil a^k/d^k \right\rceil = \left\lceil a^r/d^r \right\rceil, \text{para k } = r, r+1, ..., p-1. \quad \square$$

Lema A.7. Se $p \geq r+1$, então $\lfloor \beta^{r-1}/d^r \rfloor \leq 1$.

Demonstração: Note que se $r = \lceil d - \beta \rceil$(Lema A.4) e $d > \beta$ (equação A.3), então $r \geq 1$.

Também pelo Lema A.4, foi provado que $d^{r-1} > \beta$ e $\beta^{r-1} = \beta$. Portanto, $d^{r-1} > \beta^{r-1}$. Como $d^{r-1} = d^r + 1$, temos $d^r + 1 > \beta^{r-1}$. Dividindo-se esta inequação por d^r, obtemos

$$\frac{\beta^{r-1}}{d^r} < \frac{d^r+1}{d^r} = 1 + \frac{1}{d^r}.$$

Pela equação (2.25), $d^r \geq 1$, já que $r \leq p-1$ (premissa do lema). Então, $\beta^{r-1}/d^r < 2$, e portanto, $\lfloor \beta^{r-1}/d^r \rfloor \leq 1$. $\square$

A partir dos lemas anteriores é possível provar a Proposição 2.3, referente à forma fechada do termo a^p.

Demonstração da Proposição 2.3. Para $0 \leq p \leq r$, o Lema A.4 apresenta o resultado desta proposição. Já para $p \geq r+1$ é necessário encontrar a expressão de a^p. Através dos lemas A.4 e A.6, sabe-se que

$$\lceil a^k/d^k \rceil = \begin{cases} \lceil a/d \rceil, & \text{se } 0 \leq k \leq r-1, \\ \lceil a^r/d^r \rceil, & \text{se } r \leq k \leq p-1. \end{cases}$$

Associando este resultado ao Lema A.1, temos $\lceil a^r/d^r \rceil = \lceil a/d \rceil - \lfloor \beta^{r-1}/d^r \rfloor$. Pelo Lema A.7, $\lfloor \beta^{r-1}/d^r \rfloor = 1$. Então temos $\lceil a^r/d^r \rceil = \lceil a/d \rceil - 1$. A partir destas observações $\lceil a^k/d^k \rceil$ pode ser reescrito como

$$\lceil a^k/d^k \rceil = \begin{cases} \lceil a/d \rceil, & \text{se } 0 \leq k \leq r-1, \\ \lceil a/d \rceil - 1, & \text{se } r \leq k \leq p-1. \end{cases}$$

Utilizando os resultados acima e observando a forma recursiva do termo a^p na equação (2.24), podemos deduzir que

$$\begin{aligned} a^p &= a - \sum_{t=1}^{p} \lceil a^t/d^t \rceil \\ &= a - \sum_{t=1}^{r} \lceil a/d \rceil - \sum_{t=r+1}^{p} (\lceil a/d \rceil - 1) \\ &= a - r\lceil a/d \rceil - (p-r)(\lceil a/d \rceil - 1) \\ &= a - p\lceil a/d \rceil + (p-r). \quad \square \end{aligned}$$

A.2 Proposição 2.4

A Proposição 2.4 define os limites do termo a^p, para a positivo ou negativo. Nesta seção, os resultados desta proposição são demonstrados. Incialmente são descritos os lemas que formam base para a demonstração.

Lema A.8. $\lceil a/d \rceil = \lceil (a - a^p)/p \rceil$.

Demonstração: Como $r \geq 1$, pela Proposição 2.3, sabe-se que

$$a - p\lceil a/d \rceil \leq a^p \leq a - p\lceil a/d \rceil + (p-1),$$

e, portanto,

$$p\lceil a/d \rceil - (p-1) \leq a - a^p \leq p\lceil a/d \rceil .$$

Dividindo-se a expressão acima por p obtém-se $\lceil a/d \rceil - 1 + 1/p \leq (a - a^p)/p \leq \lceil a/d \rceil$. Como $1/p - 1 > -1$, podemos reescrever a expressão acima como

$$\lceil a/d \rceil - 1 < (a - a^p)/p \leq \lceil a/d \rceil .$$

A partir desta expressão, conclui-se que $\lceil (a - a^p)/p \rceil = \lceil a/d \rceil$. $\square$

Lema A.9. Seja $p \geq 1$, um número inteiro fixo. Seja d tal que $d \geq p$. Então, com o aumento do parâmetro d,

(i) a^p aumenta ou permanece o mesmo para $a > 0$, e

(ii) a^p diminui ou permanece o mesmo para $a \leq 0$.

Demonstração: Note que $\lceil a/d \rceil \geq 1$, para $a > 0$, e , $\lceil a/d \rceil \leq 0$, para $a \leq 0$. Suponha, primeiramente, que o termo $\lceil a/d \rceil$ continue constante, com o aumento de d. Neste caso, é necessário observar o comportamento de a^p com a variação de r. Dado que $r = \lceil d - \beta \rceil$ e $\beta = \lceil a/d \rceil d - a$, podemos reescrever r como

$$r = \lceil a - d\,(\lceil a/d \rceil - 1) \rceil . \tag{A.5}$$

Se ocorrer o diminuição de r a partir de um valor infinito, a^p permanece constante até que $p > r$ e a^p passe a agregar o termo $(p - r)$ (observe a Proposição 2.3). Neste caso a^p aumenta com a diminuição de r. A análise é análoga para o aumento de r. Assim, podemos concluir que

- Para $a > 0$, observe na equação (A.5) que o aumento de d implica que r diminui ou permanece constante, já que $\lceil a/d \rceil \geq 1$. Assim, a^p aumenta ou permanece constante.
- Para $a \leq 0$, observe na equação (A.5) que o aumento de d implica que r aumenta ou permanece constante, já que $\lceil a/d \rceil \leq 0$. Assim, a^p diminui ou permanece constante.

Suponha, agora que o termo $\lceil a/d \rceil$ varie com o aumento do parâmetro d. Então, $\lceil a/d \rceil$ diminui para $a > 0$, e aumenta para $a \leq 0$. Aplicando o Lema A.8, para p constante, pode-se concluir que

- Para $a > 0$, $\lceil a/d \rceil$ diminui,e, portanto, a^p deve aumentar para que a igualdade do Lema A.8 se mantenha.
- Para $a \leq 0$, $\lceil a/d \rceil$ aumenta,e, portanto, a^p deve diminuir para que a igualdade do Lema A.8 se mantenha. $\square$

Lema A.10. $a - \lceil a \rceil$ é o valor mínimo de a^p se $a > 0$ e o valor máximo de a^p se $a \leq 0$. Isto ocorre para $d = p$.

Demonstração: Por definição do corte CG-Nível, $d \geq p \geq 1$. Portanto $d = p$ é o valor mínimo de d. Pelo Lema A.9, para o valor mínimo de d, a^p assume seu valor mínimo, para $a > 0$ e seu valor máximo, para $a \leq 0$. Neste caso, a equação (A.5) pode ser reescrita como

$$r = \lceil a - p\,(\lceil a/p \rceil - 1) \rceil\,.$$

Como $p \in \mathbb{Z}_+$, $r = p - \lceil a/p \rceil\, p + \lceil a \rceil$, ou, $p - r = \lceil a/p \rceil\, p - \lceil a \rceil$. Vamos demonstrar que $p - r \geq 0$. Primeiramente, note que $\lceil a/p \rceil\, p - a \geq 0$. Como $\lceil a/p \rceil\, p \in \mathbb{Z}$, podemos concluir que $\lceil a/p \rceil\, p \geq \lceil a \rceil$. Logo, $p - r \geq 0$.

Portanto, se $d = p$, então $p \geq r$, e a^p assume seu valor mínimo, se $a > 0$, e seu valor máximo para $a \leq 0$. Substituindo $d = p$ na Proposição 2.3, tem-se

$$\begin{aligned} a^p &= a - p\,\lceil a/p \rceil + (p - r) \\ &= a - p\,\lceil a/p \rceil + (\lceil a/p \rceil\, p - \lceil a \rceil) \\ &= a - \lceil a \rceil\,. \quad \square \end{aligned}$$

O lema anterior define os limites mínimo e máximo para a^p, quando $a > 0$ e $a \leq 0$, respectivamente. O próximo lema afirma que é possível encontrar os outros limites para a^p, atingidos com a variação crescente do parâmetro d.

Lema A.11. O valor mínimo para a^p, para $a \leq 0$ é a, e é obtido para $d > p - a - 1$. O valor máximo a^p, para $a > 0$ é $a - p$, para $p \leq \lceil a \rceil$ e $a - \lceil a \rceil$ para $p \geq \lceil a \rceil + 1$. Este máximo é obtido para qualquer $d \geq a$.

Demonstração: Demonstra-se, inicialmente, os resultados do lema para $a \leq 0$. Considerando $d > p - a - 1$, e dividindo esta inequação por d, temos $1 > (p-1)/d - a/d$ ou $a/d > (p-1)/d - 1$. Como $d \geq p \geq 1$, temos $a/d > -1$. Portanto, $\lceil a/d \rceil = 0$ e

$$\begin{aligned} r &= \lceil d - \lceil a/d \rceil\, d + a \rceil \\ &= \lceil d + a \rceil\,. \end{aligned}$$

Como $\lceil d + a \rceil \geq p$, tem-se $p - r \leq 0$, e, pela Posposição 2.4, $a^p = a - p\,\lceil a/d \rceil = a$. Como a^p é independente de d para todo $d \geq p - a$, através do Lema A.9 conclui-se que a^p está no mínimo.

Suponha, agora, que $a > 0$. Para $d \geq a$, temos

$$\begin{aligned} r &= \lceil d - \lceil a/d \rceil\, d + a \rceil \\ &= \lceil a \rceil\,. \end{aligned}$$

Desta forma, a segunda parte do lema decorre analogamente à primeira. $\square$

Os lemas anteriores instituem premissas para a demonstração da Proposição 2.4, que trata dos limites do termo a^p em função de d.

Demonstração da Proposição 2.4: Esta proposição decorre diretamente dos lemas A.9, A.10 e A.11.

Para $a > 0$:

(i) Pelo Lema A.9, a^p é uma função não-decrescente de d.

(ii) Pelo Lema A.10, $a^p \geq a - \lceil a \rceil$.

(iii) Pelo Lema A.11, $a^p = \begin{cases} a - p, & p \leq \lceil a \rceil ; \\ a - \lceil a \rceil , & p \geq \lceil a \rceil + 1; \end{cases}$, $\forall\, d \geq a$.

Para $a \leq 0$:

(i) Pelo Lema A.9, a^p é uma função não-crescente de d.

(ii) Pelo Lema A.10, $a^p \leq a - \lceil a \rceil$.

(iii) Pelo Lema A.11, $a^p = a$, $\forall\, d \geq p - a$.

Os resultados da proposição decorrem diretamente. □

A.3 Proposição 2.5

Esta proposição mostra quais as restrições sobre p, d e y, para que $a^p = y$. Pela própria definição do corte CG-Nível, temos $p \in \mathbb{Z}$. Pela equação (2.25) , temos $d \geq p \geq 1$.

Pela definição de a^p na Proposição 2.3, se $a^p = y$, então tem-se

$$a - a^p = a - y = \begin{cases} p \lceil a/d \rceil , & \text{se } 1 \leq p \leq r, \\ p \lceil a/d \rceil + (p - r), & \text{se } p \geq r + 1. \end{cases}$$

Portanto $a - y \in \mathbb{Z}$, já que p, $\lceil a/d \rceil$ e r são inteiros. Como já definido na Proposição 2.5, temos:

$$\alpha = \lceil (a - y)/p \rceil - 1. \tag{A.6}$$

Esta definição facilita a determinação dos intervalos válidos para d, bem como a presente demonstração. Pelo Lema A.8, se $a^p = y$, então tem-se

$$\lceil a/d \rceil = \lceil (a - y)/p \rceil \text{ , ou,}$$
$$\lceil a/d \rceil = \alpha + 1. \tag{A.7}$$

Portanto temos $\alpha < a/d \leq \alpha + 1$, ou

$$d\alpha < a \leq d(\alpha + 1). \tag{A.8}$$

Observando a definição do termo a^p na Proposição 2.3, nota-se que a diferença entre as duas possíveis equações é o termo $(p-r)$. Assim, podemos reescrever y como

$$y = a - p\lceil a/d \rceil + (p-r)^+. \tag{A.9}$$

Esta equação motiva a definição de δ, de modo que

$$\delta = (p-r)^+ = p\lceil a/d \rceil - (a-y).$$

Pelo Lema A.8,

$$\delta = p\lceil (a-y)/p \rceil - (a-y).$$

Observando a definição de α na equação (A.6) , é possível chegar à definição de δ que é colocada no enunciado desta proposição. Assim, temos

$$\delta = p(\alpha+1) - (a-y). \tag{A.10}$$

A equação (A.9) pode ser reescrita de forma mais conveniente como $y = a - p\lceil a/d \rceil + \delta$, dado que existe um d que satisfaça (A.8). Assim, $a^p = y$, se e somente se

$$\begin{aligned} &p - r = \delta, \quad \delta > 0, \\ &p - r \leq \delta, \quad \delta = 0. \end{aligned}$$

Em ambos os casos, $p - r \leq \delta$. Mas $r = \lceil d - \beta \rceil < d - \beta + 1$. Portanto, $p - \delta \leq r < d - \beta + 1$, e , assim, $p - \delta < d - \beta + 1$. Pela definição de β na equação (A.1), temos $d - \beta = d - (\lceil a/d \rceil\, d - a)$. Substituindo (A.7), temos $d - \beta = d - ((\alpha+1)d - a) = a - d\alpha$. Assim, temos

$$d\alpha < a + \delta - p + 1. \tag{A.11}$$

Quando $\delta = 0$, as equações (A.8) e (A.11) são condições necessárias e suficientes para garantir que $a^p = y$. Para $\delta > 0$, deve ser adicionada a condição $p - r \geq \delta$, para que se tenha $p - r = \delta$.

Se $p - r \geq \delta$, então $p - \delta \geq r \geq \lceil d - \beta \rceil$. Mas $p - \delta \geq \lceil d - \beta \rceil$, se e somente se, $p - \delta \geq d - \beta$, já que p e δ são inteiros. A mesma análise aplicada à equação (A.11) pode ser repetida aqui, originando

$$d\alpha \geq a - p + \delta. \tag{A.12}$$

O limitante superior para $d\alpha$ dado pela equação (A.8) é maior que aquele imposto por (A.11), pois $\delta \leq p - 1$.

O limitante inferior para $d\alpha$ é dado apenas pela equação (A.8), se $\delta = 0$. Se $\delta > 0$, o limitante inferior é o máximo entre (A.8) e (A.12). Para que o limitante dado por (A.8) seja maior que o dado por (A.12), devemos ter $d + \delta - p \geq 0$, que segue diretamente $d \geq p$ e $\delta \geq 0$.

Nas equações (A.11) e (A.12), o termo δ pode ser substituído segundo a equação (A.10). Assim, estas equações podem ser reescritas como

$$d\alpha < p\alpha + y + 1, \tag{A.13}$$

e,

$$d\alpha \geq p\alpha + y. \tag{A.14}$$

Portanto, as condições para que $a^p = y$ podem ser resumidas como

$$a^p = y \Leftrightarrow \begin{cases} p\alpha + y \leq d\alpha < p\alpha + y + 1, & \text{se } \delta > 0, \\ d(\alpha+1) \geq a, \text{ e, } d\alpha < p\alpha + y + 1, & \text{se } \delta = 0. \end{cases}$$

Note que, se $a > 0$, então $\lceil a/d \rceil \geq 1$, e portanto, pela equação (A.7),

$$\alpha \geq 0. \tag{A.15}$$

Para $a \leq 0$, $\lceil a/d \rceil \leq 0$, e portanto, pela equação (A.7),

$$\alpha \leq -1. \tag{A.16}$$

Os resultados da proposição decorrem diretamente destas observações.

A.4 Proposição 2.7

Para esta demonstração são definidos, dado quaisquer p e d, tais que $d \geq p$, os novos parâmetros p' e d' dados por

$$p' = p + \Delta_p, \text{ e,}$$
$$d' = d + \Delta_d,$$

onde $\Delta_p \geq \Delta_d \geq 0$ e Δ_p é inteiro. Para manter a coerência do corte CG-Nível, as variações em p e d devem respeitar a restrição de que $d' \geq p'$.

Sejam α e δ definidos segundo as equações (A.6) e (A.10). Substituindo y por a^p, temos

$$a^p = a - p(\alpha + 1) + \delta.$$

Sejam α' e δ', definidos analogamente, relativos a p', d' e $a^{p'}$. Assim, temos

$$a^{p'} = a - p'(\alpha' + 1) + \delta'. \tag{A.17}$$

Definindo $\Delta' = \alpha - \alpha'$ e subtraindo as duas equações anteriores, temos

$$\delta' = \delta - p\Delta' + \Delta p(\alpha' + 1) + a^{p'} - a^p. \tag{A.18}$$

Inicialmente é provada por contradição a condição (i), para $a > 0$. Para tanto, suponha $a^{p'} - a^p > 0$. Como esta diferença é inteira, $a^{p'} - a^p \geq 1$. Pelas definições de α e α' e observando o resultado do Lema A.8, temos

$$\Delta' = \alpha - \alpha' = \lceil a/d \rceil - \lceil a/d' \rceil \geq 0, \tag{A.19}$$

já que $d' \geq d$. Além disso, com $\delta \geq 0$(Proposição 2.5) e $\alpha' \geq 0$(equação (A.15)), a partir da expressão de δ' (A.18) é possível concluir que $\Delta' = 0$ implica em $\delta' > 0$, já que supomos que $a^{p'} - a^p \geq 1$.

Pela inequação (A.13), substituindo $y = a^p$, tem-se $d\alpha < p\alpha + a^p + 1$. Mas $p\alpha = (p' - \Delta p)(\Delta' + \alpha') = p'\alpha' + p'\Delta' - \alpha\Delta p$. Analogamente, $d\alpha = d'\alpha' + d'\Delta' - \alpha\Delta d$. Assim, a inequação (A.13) pode ser reescrita como

$$p'\alpha' + \alpha(\Delta_d - \Delta_p) - (d' - p')\Delta' + a^p + 1 > d'\alpha'. \tag{A.20}$$

Suponha que $\delta' > 0$. Então, pela expressão (A.14), temos $d'\alpha' \geq p'\alpha' + a^{p'}$, que pode ser substituída em (A.20), originando

$$\alpha(\Delta_d - \Delta_p) > (a^{p'} - a^p - 1) + (d' - p')\Delta'. \tag{A.21}$$

Por hipótese $a^{p'} - a^p \geq 1$. Mas, como já colocado, $\Delta' > 0$, $(d' - p') \geq 0$, $\alpha \geq 0$ e $\Delta_d - \Delta_p \leq 0$. Portanto, a inequação acima resulta em $0 > 0$ e fica provado por contradição que (i) é verdadeiro para $\delta' > 0$.

Suponha, agora, que $\delta' = 0$. A partir da inequação (A.8) temos $d'\alpha' \geq a - d'$. Como $\delta' = 0$, a inequação (A.17) pode ser reescrita como $a^{p'} = a - p'(\alpha' + 1)$, e, portanto, $a = a^{p'} + p'\alpha' + p'$. Substituindo esta expressão em $d'\alpha' \geq a - d'$ e aplicando (A.20), obtém-se

$$\alpha(\Delta_d - \Delta_p) > (a^{p'} - a^p - 1) + (\Delta' - 1)(d' - p'). \tag{A.22}$$

Pela equação (A.19), $\Delta' \geq 0$. Mas $\Delta' = 0$ implica $\delta' > 0$, como mostrado anteriormente a partir da análise de (A.18). Como estamos supondo $\delta' = 0$, temos $\Delta' \geq 1$. Analogamente à dedução feita para a expressão (A.21), estas hipóteses, se aplicadas à inequação acima, levam à contradição $0 > 0$. Desta forma, (i) fica demonstrado.

A demonstração de (ii) também é feita por contradição. Portanto, assume-se que $a^p - a^{p'} \geq 1$. Da mesma forma que foi feito para $a > 0$, pode-se derivar que

$$\alpha(\Delta_p - \Delta_d) > (a^p - a^{p'} - 1) - (d' + p')\Delta', \text{ para } \delta' > 0, \tag{A.23}$$

e

$$(\alpha + 1)(\Delta_p - \Delta_d) > (a^p - a^{p'} - 1) - (\Delta' + 1)(d' - p'), \text{ para } \delta' = 0. \tag{A.24}$$

Pela inequação (A.16), para $a \leq 0$, $\alpha \leq -1$. Como $\Delta p \geq \Delta d$, o lado esquerdo de (A.23) e (A.24) é ≤ 0. Aplicando a mesma análise feita a partir de (A.18) e (A.19), conclui-se que $\Delta' \leq 0$ e que $\Delta' = 0$ implica $\delta' > 0$, e então $\delta = 0$ implica $\Delta' \leq -1$. Substituindo estes valores nas duas inequações acima, são geradas contradições do tipo $0 > 0$, e o resultado da proposição segue diretamente.