Este exemplar corresponde à redação final da Tese/Dissertação devidamente corrigida e defendida por: Felipe Carneiro Calheiros
e aprovada pela Banca Examinadora.
Campinas, 22 de janeiro de 2002
COORDENADOR DE PÓS-GRADUAÇÃO
CPG-IC

Partições Retangulares Ótimas: Algoritmos Lagrangeanos e Planos de Corte

Felipe Carneiro Calheiros

Dissertação de Mestrado

UNICAMP
BIBLIOTECA CENTRAL
SEÇÃO CIRCULANTE

Instituto de Computação
Universidade Estadual de Campinas

Partições Retangulares Ótimas: Algoritmos Lagrangeanos e Planos de Corte

Felipe Carneiro Calheiros[1]

Agosto de 2001

Banca Examinadora:

- Prof. Dr. Cid Carvalho de Souza (Orientador)
- Prof. Dr. Henrique Pacca Loureiro Luna
  Departamento de Ciência da Computação – UFMG
- Prof. Dr. Flávio Keidi Miyazawa
  Instituto de Computação – UNICAMP
- Prof. Dr. Pedro Jussieu de Rezende
  Instituto de Computação – UNICAMP

[1]Parcialmente apoiado pelo CNPQ.

UNICAMP
BIBLIOTECA CENTRAL

UNIDADE BC
Nº CHAMADA T/UNICAMP
C128p
V
TOMBO BC/ 48163
PROC. 16-837/02
C
PREÇO R$ 11,00
DATA
Nº CPD

CM00165785-0

BCD 235898

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Calheiros, Felipe Carneiro

C128p Partições retangulares ótimas: algoritmos lagrangeanos e planos de corte / Felipe Carneiro Calheiros -- Campinas, [S.P. :s.n.], 2001.

Orientadores: Cid Carvalho de Souza; Abílio Pereira de Lucena Filho

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Programação inteira. 2. Otimização combinatória. 3. Programação linear. I. Souza, Cid Carvalho de. II. Lucena Filho, Abílio Pereira de. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Partições Retangulares Ótimas: Algoritmos Lagrangeanos e Planos de Corte

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Felipe Carneiro Calheiros e aprovada pela Banca Examinadora.

Campinas, 17 de Setembro de 2001.

Prof. Dr. Cid Carvalho de Souza
(Orientador)

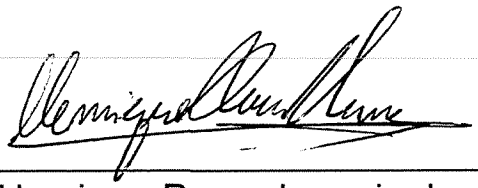
Prof. Dr. Abílio Pereira de Lucena Filho
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

UNICAMP

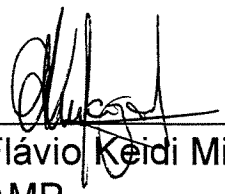
TERMO DE APROVAÇÃO

Tese defendida e aprovada em 14 de setembro de 2001, pela Banca Examinadora composta pelos Professores Doutores:



Prof. Dr. Henrique Pacca Loureiro Luna
UFMG

Prof. Dr. Pedro Jussieu de Rezende
IC - UNICAMP



Prof. Dr. Flávio Keidi Miyawawa
IC - UNICAMP

Prof. Dr. Cid Carvalho de Souza
IC - UNICAMP

© Felipe Carneiro Calheiros, 2001.
Todos os direitos reservados.

Agradecimentos

Agradeço...

Aos meus pais, pelo apoio, incentivo, carinho, amizade e, principalmente, pelo amor incondicional.

À minha mãe, especialmente, porque em muitos momentos abdicou da sua própria vida em favor da minha.

Ao meu pai, especialmente, por sempre ter movido montanhas para possibilitar que eu aproveitasse as grandes oportunidades da minha vida.

Às minhas irmãs por compartilharem comigo os momentos felizes de minha vida, em especial, a alegria da conclusão deste trabalho.

Aos meus orientadores, Cid e Abílio, pela amizade e confiança. Ainda, pelos exemplos de perseverança e dedicação à pesquisa científica. Exemplos que me mostraram que, mesmo com dificuldades, ainda é possível fazer um trabalho sério e honesto.

Ao professor Pedro Jussieu de Rezende, pela contribuição extremamente importante na minha formação acadêmica.

À Vera e Daniel, por sempre responderem com sorriso e eficiência às muitas solicitações venho fazendo desde os meus tempos de graduando.

Aos amigos do GOA, Tallys e Maikol, pela disposição sempre constante de ajudar.

Aos amigos Ricardo, Virgílio, Carlos e Renata que sempre me receberam muito bem em sua casa, nas minhas idas a Campinas.

Aos demais amigos que, de diferentes maneiras, tornaram cada dia mais interessante.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, pelo apoio financeiro.

É melhor tentar e falhar,
Que preocupar-se e ver a vida passar.
É melhor tentar ainda em vão,
Que sentar-se fazendo nada até o final.
Eu prefiro na chuva caminhar,
Que em dias tristes em casa me esconder.
Prefiro ser feliz, embora louco,
Que em conformidade viver.

Martin Luther King

O infinito é realmente
um dos deuses mais lindos.

Renato Russo

Resumo

Seja P um conjunto finito de pontos do plano localizados no interior de um retângulo R. Considere as partições de R em retângulos menores. Se nenhum ponto de P for interior a algum destes retângulos, então a partição é viável e seu custo é a soma do comprimento dos segmentos que a definem. O problema de partição retangular (**RG-P**) busca uma partição retangular de R de custo mínimo. Experimentos descritos na literatura envolvendo algoritmos exatos para esse problema indicam que instâncias do **RG-P** com pontos não corretilineares em P, chamado de **RG-NLP**, são as mais difíceis de serem resolvidas até a otimalidade. Esse trabalho apresenta propriedades geométricas de soluções ótimas do **RG-NLP**, que permitem uma redução substancial do número de variáveis do modelo natural de programação inteira. Adicionalmente, essas propriedades levam a desigualdades satifeitas por todas as soluçoes ótimas. Tais desigualdades são usadas em um algoritmo de Relaxação Lagrangeana com Planos de Corte (*Relax and Cut*). Enquanto resolvedores de programação linear não conseguem computar o enorme modelo para instâncias do **RG-NLP**, o algoritmo lagrangeano produziu excelentes limitantes. Além disso, para as instâncias em que este algoritmo não foi capaz de provar otimalidade, o número de variáveis eliminadas durante a sua execução foi suficiente para permitir a execução do resolvedor de programação linear. O algoritmo híbrido combinando a Relaxação Lagrangeana e Programação Linear foi capaz de resolver instâncias mais do que duas vezes maiores que as apresentadas na literatura. Além disso, os grandes modelos de partição gerados e resolvidos neste trabalho figuram entre os maiores já resolvidos até a otimalidade.

Abstract

Let P be a finite set of points in the plane lying in the interior of a rectangle R. Consider the partitions of R into smaller rectangles. If no point in P is interior to any such rectangle, the partition is feasible and its length is the sum of the lengths of the segments defining it. The Rectangular Partition Problem (**RG-P**) seeks for a feasible partition of R of minimum length. Experiments reported in the literature with exact algorithms based on Integer Programming (IP) indicate that **RG-P** instances with non corectilinear points in P, called **RG-NLP**, are the hardest to solve to optimality. This work presents structural properties of optimal **RG-NLP** solutions which allow for substantial reductions on the number of variables in the natural IP model of the problem. In addition, these properties led to inequalities that are satisfied by all optimal solutions. Such inequalities are used in a Lagrangean Relax and Cut algorithm for **RG-NLP**. While commercial LP solvers cannot compute the huge models for **RG-NLP** instances in general, the Lagrangean algorithm produces very good bounds. For the few test instances where Lagrangean bounds alone are not enough to prove optimality, they allow enough variables to be fixed that an LP solver can now be applied. The hybrid algorithm combining the Lagrangean and the LP phases solves **RG-NLP** instances more than twice as large as those in the literature. Additionally, the large set partitioning instances solved with this algorithm figure among the biggest ever solved to prove optimality.

Sumário

Agradecimentos xi

Resumo xv

Abstract xvii

Lista de Tabelas xxi

Lista de Figuras xxiii

1 Introdução 1

1.1 Motivação 3

1.2 Objetivos do Trabalho 4

1.3 Organização do Texto 4

2 Fundamentos Teóricos 5

2.1 Combinatória poliédrica 6

2.2 Relaxação Lagrangeana e Planos de Corte (*Relax & Cut*) 7

2.2.1 Relaxação Lagrangeana 7

2.2.2 Otimização por Subgradiente 8

2.2.3 Heurística Lagrangeana e Redução do Problema 10

2.2.4 Planos de Corte 11

2.3 Conjuntos Independentes 12

2.3.1 Desigualdades de Clique 13

2.3.2 Desigualdades de Ciclo Ímpar 14

3 Formulação do Problema RG-P 17

3.1 Definições e Resultados Básicos 17

3.2 Uma Formulação de PLI para o **RG-P** 19

4 Reduções Geométricas e Novas Desigualdades Válidas **23**
4.1 Retângulos com Ponto Terminal no Vértice 24
4.2 Retângulos com Interseção nos Lados Estendidos 25
4.3 Retângulos Canônicos com Pontos Terminais nos Vértices 27
4.4 Desigualdades Válidas 29
4.4.1 Cliques e Ciclos Ímpares 29
4.4.2 Restrição de Cardinalidade 30

5 O Algoritmo **33**
5.1 Uma Visão Geral do Algoritmo 33
5.2 O Problema de Relaxação Lagrangeana para o **RG-NLP** 34
5.3 O Método do Subgradiente 36
5.4 A Heurística Primal 37
5.5 Geração de Desigualdades Válidas 37
5.6 Fixação de Variáveis 39
5.7 Tópicos de Implementação 41

6 Experimentos Computacionais **45**
6.1 Instâncias 45
6.2 Resultados das Reduções Geométricas 46
6.3 Resultados Obtidos com o CPLEX 48
6.4 Resultados Obtidos com a Heurística Lagrangeana 50
6.4.1 Heurística Lagrangeana sem Cortes 50
6.4.2 Heurística Lagrangeana com Cortes 53
6.4.3 Mudando os Parâmetros de Execução 57
6.4.4 *ReStart* 60
6.4.5 Instâncias com *GAP* de Integralidade 62

7 Conclusão e Trabalhos Futuros **63**

A Algoritmos de Separação de Desigualdade Válidas **65**

B Novos Resultados **67**

C Exemplos de soluções ótimas **71**

Bibliografia **75**

Lista de Tabelas

6.1 Redução de colunas. Instâncias de pequeno e médio porte. 47
6.2 Redução de colunas. Instâncias de grande porte. 48
6.3 Resultados obtidos com o CPLEX 49
6.4 Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes. . . . 51
6.5 Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes. Instâncias de grande porte. 52
6.6 Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana sem cortes 53
6.7 Resultados obtidos com a Heurística Lagrangeana com o uso de cortes. . . . 54
6.8 Resultados obtidos com a Heurística Lagrangeana com o uso de cortes. Problemas de grande porte. 55
6.9 Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana com cortes 55
6.10 Cortes gerados durante a execução da heurística lagrangeana. 56
6.11 Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes com γ=300 e 8000 iterações. 58
6.12 Resultados obtidos com a Heurística Lagrangeana com o uso de cortes com γ=300 e 8000 iterações. 59
6.13 Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana sem cortes com γ=300 e 8000 iterações. 59
6.14 Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana com cortes com γ=300 e 8000 iterações. 59
6.15 Resultados obtidos com a Heurística Lagrangeana com cortes e uso da técnica de *ReStart*. 61
6.16 Resultados médios obtidos para 18 instâncias de 22 pontos com GAP de integralidade. 62

B.1 Redução de colunas. Instâncias acima de 100 pontos. 68

UNICAMP
BIBLIOTECA CENTRAL

B.2 Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes com γ=300 e 8000 iterações. Instância acima de 100 pontos 68
B.3 Resultados obtidos com a Heurística Lagrangeana com o uso de cortes com γ=300 e 8000 iterações. Instâncias acima de 100 pontos 69
B.4 Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana sem cortes com γ=300 e 8000 iterações. Instâncias acima de 100 pontos. . 69
B.5 Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana com cortes com γ=300 e 8000 iterações. Instâncias acima de 100 pontos. . 69

Lista de Figuras

1.1 Uma instância do **RG-NLP**. 2
1.2 Uma solução viável para a instância da figura 1.1. 2

2.1 Exemplos de grafos suporte para geração de desigualdades de clique e ciclo ímpar. 14

3.1 Uma instância $I = (R, P)$ do **RG-P**. Os segmentos pontilhados representam **GI(P)**, os pontos cheios são terminais e os brancos são pontos de Steiner. . . 18
3.2 (a) Uma Instância do **RG-P**. (b) Retângulos canônicos em **GI(P)**. (c) Um retângulo viável. 20

4.1 Possibilidades de movimento de segmentos de soluções viáveis visando diminuir o valor da solução. Os círculos representam os pontos terminais e os losangos, os pontos de Steiner. 24
4.2 Seis modos de um ponto ter grau 2. Em dois deles os segmentos incidentes pertencem à mesma reta e nos quatro restantes, esses segmentos são perpendiculares. 25
4.3 (a) Uma instância do **RG-NLP** e suas linhas de grade. (b) Os lados estendidos do retângulo R_j. 27
4.4 Um ponto terminal e seus quatro retângulos canônicos. 28
4.5 Um exemplo onde os retângulos R_i e R_j possuem interseção estendida. . . . 30

5.1 Fluxograma de execução da Heurística Lagrangeana com inserção de planos de corte. 35
5.2 Modelo de classes de problemas de otimização combinatória usando algoritmos lagrangeanos. 41

6.1 Variáveis em função do número de pontos da instância, comparado com $|P|^3$ 46
6.2 Número de desigualdades ativas ao longo das iterações. As linhas verticais representam o ponto onde o parâmetro α é reduzido. (a) Instância com 17 pontos. (b) Instância com 58 pontos. 56

6.3 (a) Tempo em função do número de pontos da instância. (b) Tempo em função do número de variáveis. 57

C.1 Solução ótima para instância com 15 pontos. 71
C.2 Solução ótima para instância com 17 pontos. 72
C.3 Solução ótima para instância com 46 pontos. 73
C.4 Solução ótima para instância com 80 pontos. 74

Capítulo 1

Introdução

Neste trabalho será estudado um caso particular do problema conhecido como *Problema de Partição Retangular de um Retângulo com Pontos no Interior* (**RG-P**). Primeiro será definido o **RG-P** e posteriormente será enunciado o caso particular.

Uma instância $I = (R, P)$ do **RG-P** é caracterizada por um retângulo R, também chamado de retângulo envoltório, e por um conjunto de pontos P localizados no interior de R. Uma partição retangular de R com relação a P é caracterizada por um conjunto de segmentos S no interior de R, que o particiona em retângulos menores de tal modo que todo ponto de P faz parte de um segmento de S. O problema **RG-P** consiste em minimizar a soma dos comprimentos dos segmentos de S.

Este trabalho se concentrará na resolução do **RG-NLP**, que é um caso particular do **RG-P**, no qual o conjunto P não possui pontos corretilineares, ou seja, na mesma reta horizontal ou vertical. O motivo do estudo deste caso específico é que trabalhos anteriores mostraram que a sua resolução é computacionalmente muito mais custosa que o caso com pontos corretilineares. Em [9], foram resolvidos até a otimalidade problemas com até 200 pontos no caso do **RG-P** e somente 39 pontos para o **RG-NLP**. Essa diferença pode ser explicada pelo fato de o modelo utilizado gerar mais variáveis e restrições para o **RG-NLP** do que para o **RG-P**, como será explicado no capítulo 3. As figuras 1.1 e 1.2 mostram respectivamente uma instância do **RG-NLP** e uma de suas soluções viáveis.

Finalmente, em [1] pode-se verificar que o **RG-P** é um problema NP-difícil. O mesmo ainda não foi provado para o **RG-NLP**.

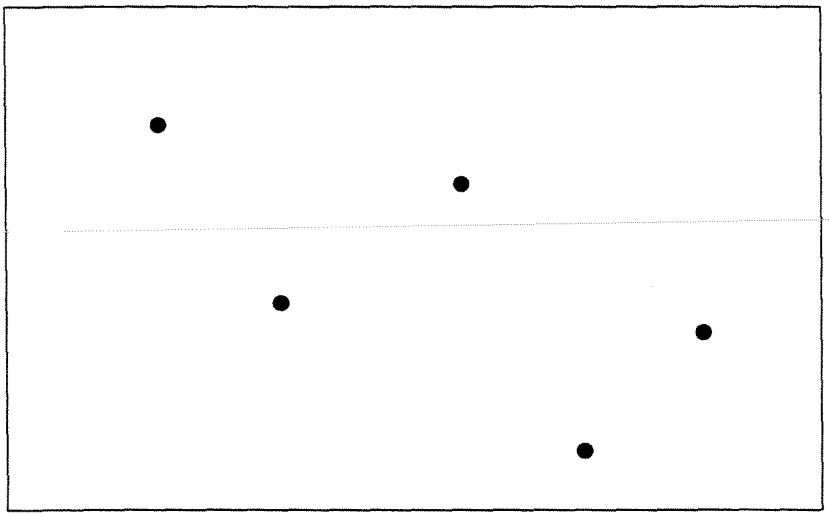


Figura 1.1: Uma instância do **RG-NLP**.

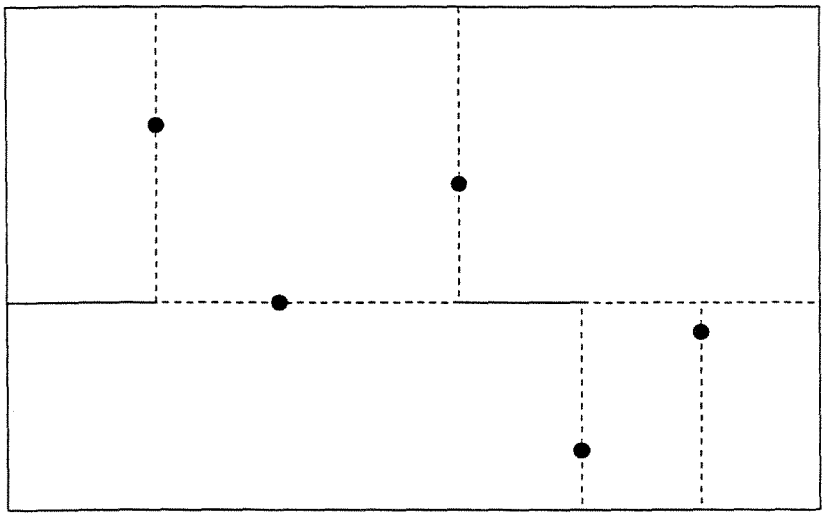


Figura 1.2: Uma solução viável para a instância da figura 1.1.

1.1 Motivação

A aplicação que motiva o estudo deste problema se encontra no projeto de circuitos integrados. Abaixo segue um resumo desta aplicação baseada em [1, 2].

O projeto do posicionamento e interconexão para circuitos integrados (CI) é dividido em alguns passos. Dentre eles tem-se o planejamento da rede de caminhos dos sinais, que vem após o posicionamento dos módulos de lógica e do projeto dos barramentos de força e aterramento. O planejamento da rede de caminhos de sinais começa com a definição dos canais (*channels*). No procedimento de definição de canais, a área da placa (*chip*) fora dos módulos de lógica é particionada em retângulos sem interseção (canais) dentro dos quais os circuitos serão feitos posteriormente.

O sistema de posicionamento e interconexão assume neste ponto que existe uma única camada de metal, e que as redes de força e aterramento já estão planejadas nesta camada antes da definição das redes de sinais. Desta forma, ao final do procedimento de definição de canais a área do chip estará dividida em retângulos de três tipos:

- Módulos de lógica
- Regiões livres de lógica
- Regiões cobertas por metal (abaixo de um barramento de força ou terra), mas livre em outras camadas. Essas regiões são conhecidas como canais descobertos.

Os canais são separados uns dos outros e dos módulos de lógica por arestas verticais ou horizontais. Cada uma destas arestas está na borda de exatamente dois canais ou módulos distintos, a menos que pertençam à borda da placa do circuito integrado.

A descrição em [2] diz que o objetivo desta fase é obter canais "grandes e de aparência natural", o que pode ser obtido, de acordo com o autor, minimizando a soma dos custos das arestas.

É interessante notar que a aplicação descrita acima difere daquela do início do capítulo, uma vez que nela temos buracos ao invés de pontos. Apesar disto, como esses buracos têm a forma de polígonos retilineares, os resultados da resolução do **RG-P** podem ser facilmente estendidos para adaptar à versão com buracos. Basta para isso colocar um ponto em cada vértice de cada polígono que representa um buraco.

Esse problema atraiu pesquisadores da área de geometria computacional e uma série de trabalhos foi desenvolvida sobre o tema. Em [3, 4, 5, 6, 7, 8] encontram-se algoritmos de aproximação para o problema que utilizam técnicas como divisão e conquista e programação dinâmica. Em [9, 10] foram desenvolvidos algoritmos exatos utilizando programação linear inteira (Branch & Bound, Branch & Cut e Branch & Price). Estes últimos trabalhos propuseram uma modelagem do **RG-P** como um problema de partição de conjuntos que será também utilizada neste trabalho.

1.2 Objetivos do Trabalho

Os objetivos principais deste trabalho são:

- Investigar propriedades geométricas que permitam a redução do número de variávies do modelo de retângulos descritos em [9].
- Encontrar desigualdades válidas para este modelo.
- Desenvolver um procedimento baseado em relaxação lagrangeana que se utilize dos resultados dos tópicos anteriores para obter uma solução heurística de boa qualidade (ótima quando possível) para instâncias maiores do **RG-NLP**.

1.3 Organização do Texto

O texto deste trabalho está organizado da seguinte maneira. No capítulo 2 são apresentadas as bases teóricas que darão suporte ao trabalho. No capítulo 3 é apresentada a formulação do modelo de retângulos para o **RG-P** (proposta em [9]). No capítulo 4 são provadas algumas propriedades geométricas que levam à redução no número de variáveis e restrições do modelo original. Neste capítulo ainda são descritas novas desigualdades válidas para o problema. O capítulo 5 apresenta uma discussão pormenorizada do algoritmo implementado. O capítulo 6 mostra os resultados obtidos e, por fim, o capítulo 7 apresenta as conclusões deste trabalho e aponta novas direções de pesquisa com vistas à obtenção de melhores soluções para este problema.

Capítulo 2

Fundamentos Teóricos

Neste capítulo serão apresentadas introduções dos fundamentos teóricos utilizados neste trabalho. A leitura deste capítulo proverá o conhecimento básico dos conceitos abordados, no entanto, para um aprofundamento maior nos diversos tópicos aqui discutidos, recomenda-se a leitura das citações fornecidas no decorrer do texto.

Inicialmente serão apresentados conceitos básicos de combinatória poliédrica. Em seguida apresentamos as técnicas de relaxação lagrangeana, algoritmos de planos de corte facial e, finalmente, o capítulo é encerrado com exemplos de planos de corte obtidos para o problema dos conjuntos independentes.

Considere, então, o seguinte problema de programação linear:

$$min\{cy \mid Ay \leq b, y \in \mathbb{R}^n\}, \tag{2.1}$$

onde $c \in \mathbb{R}^n$, A é uma matriz $m \times n$ e $b \in \mathbb{R}^n$. Se as variáveis forem restritas a valores inteiros, o problema se torna um problema de Programação Linear Inteira (PLI). Se as variáveis forem restritas ainda mais, só sendo permitidos os valores 0 ou 1, então temos um problema de PLI 0-1.

Problemas de PLI 0-1 são, geralmente, NP-difíceis. Desta maneira, os algoritmos desenvolvidos para sua resolução atacam problemas gerados a partir do problema original cujas soluções fornecem limites inferiores (para o caso de problemas de minimização) que são mais facilmente encontradas. Essas formas mais simples são chamadas de relaxações. A relaxação mais conhecida é a linear, onde as restrições de integralidade são substituídas por restrições lineares de pertinência ao intervalo $[0, 1]$. Existem algumas abordagens clássicas que utilizam relaxação linear para resolver problemas de PLI 0-1, são elas: algoritmos de planos de corte, algoritmos de *branch and bound* e ainda, a combinação dos dois, o *branch and cut*. Outro tipo é a Relaxação Lagrangeana. Esta será objeto do nosso estudo e será mais detalhada na seção 2.2.

2.1 Combinatória poliédrica

Nesta seção serão apresentados alguns conceitos e definições básicos de combinatória poliédrica que serão úteis no decorrer do texto. Uma abordagem mais completa e formal pode ser encontrada em [21], [22] ou [23].

Considere o problema de programação linear descrito em 2.1. O conjunto convexo $Y = \{y \in \mathbb{R}^n | Ay \leq b\}$ é chamado de poliedro. Caso Y seja limitado, isto é, exista um valor $w \in \mathbb{R}_+$ tal que $\{-w \leq y_j \leq w, \forall y_j \in Y\}$, então este é também chamado de politopo.

Um conjunto Y é convexo se dados dois elementos y_1 e $y_2 \in Y$ todos os pontos da forma $\alpha y_1 + (1-\alpha)y_2$, $0 \leq \alpha \leq 1$, também pertecem a Y. Os pontos obtidos desta forma são combinações convexas de y_1 e y_2. O conjunto gerado pela combinação convexa de todos os subconjuntos finitos de Y é denominado envoltória convexa de Y, denotada por conv(Y). Em outras palavras, conv(Y) é o menor conjunto convexo que contém Y. A envoltória convexa de um poliedro tem algumas propriedades importantes, são elas: (i) conv(Y) é um politopo e (ii) os pontos extremos de conv(Y) também pertencem a Y.

Usando essas duas propriedades é possível verificar que a resolução do problema de PLI dado por $min\{cy \mid y \in Y\}$ é equivalente ao problema de programação linear dado por $min\{cy \mid y \in \text{conv}(Y)\}$. Essa conclusão proveria um excelente método para a resolução do problema inteiro. No entanto, em muitos casos existe um número muito grande (exponencial no número de variáveis) de desigualdades necessárias para descrever conv(Y). Assim, uma melhor estratégia é buscar métodos que levem à aproximação de conv(Y). Nesse sentido serão definidos a seguir os conceitos de desigualdades válidas e facetas de um poliedro.

Definição 1 *Uma desigualdade $ay \leq b$ é válida para $Y \subseteq \mathbb{R}^n$ se $ay_j \leq b$, $\forall y_j \in Y$.*

A partir da definição acima é fácil ver que tanto as desigualdades originais do problema, como as que descrevem a envoltória convexa são necessariamente válidas para Y.

A definição de desigualdade válida leva a uma questão bastante interessante. Quais dessas desigualdades são úteis? A resposta para essa pergunta virá da definição do conceito de dominância que se segue.

Definição 2 *A desigualdade $ay \leq b$ domina $cy \leq d$ se $\{y \in \mathbb{R}^n_+ \mid ay \leq b\} \subseteq \{y \in \mathbb{R}^n_+ \mid cy \leq d\}$.*

Outra forma de uma desigualdade ser dominada, e portanto redundante, é ser dominada por uma combinação linear de duas ou mais desigualdades válidas.

Finalmente, define-se uma faceta de um poliedro como uma desigualdade que não é dominada por nenhuma outra. Sendo assim, facetas são as desigualdades mais fortes que podem ser definidas para um poliedro. Métodos formais de provar que uma desigualdade caracteriza uma faceta podem ser encontrados em [21].

2.2 Relaxação Lagrangeana e Planos de Corte (*Relax & Cut*)

Esta seção é o resumo de uma pesquisa baseada em alguns artigos e livros. Para os tópicos Relaxação Lagrangeana e Otimização por Subgradiente recorremos a [12, 13, 14]. O artigo de Beasley [14] também é uma boa fonte de leitura para Heurística Lagrangeana e Redução do Problema através da fixação de variáveis. Finalmente, os trabalhos de [15, 16, 17, 23] fornecem o material necessário para a discussão de planos de corte.

2.2.1 Relaxação Lagrangeana

Os problemas abordados em otimização combinatória geralmente são muito difíceis de resolver. Como mencionado no início do capítulo eles são normalmente NP-difíceis e, a menos que $P = NP$, só existem algoritmos exponenciais para a sua resolução. Entretanto, alguns desses problemas podem ser vistos como problemas fáceis complicados por algum conjunto de restrições. A pergunta que surge naturalmente é: como aproveitar esta característica para desenvolver um algoritmo para o problema? Uma possível resposta vem do trabalho de Held e Karp [12]. A técnica é chamada de Relaxação Lagrangeana e é usada para gerar limites inferiores que podem ser utilizados em vários algoritmos combinatórios.

A Relaxação Lagrangeana consiste em passar para a função objetivo, com um fator de penalização, um certo subconjunto das restrições do problema original. Diz-se que estas restrições foram dualizadas. Abaixo segue um exemplo que auxiliará a compreensão da técnica.

Considere o seguinte problema **P** de programação linear inteira.

$$\begin{aligned} Z &= \min cy \\ &s.a. \quad Ay \geq b \qquad (2.2) \\ &\quad\; Dy = e \qquad (2.3) \\ &\quad\; y \geq 0,\ y \in \mathbb{Z} \end{aligned}$$

Os fatores de penalização são chamados de multiplicadores de Lagrange e são denotados por λ. Então, introduzindo no exemplo acima o vetor $\lambda \geq 0$ e dualizando o conjunto de restrições 2.2, obtem-se o problema relaxado $\mathbf{LD}_\lambda$ dado por:

$$\begin{aligned} Z_D(\lambda) &= \min cy + \lambda(b - Ay) \\ &s.a. \quad Dy = e \\ &\quad\; y \geq 0,\ y \in \mathbb{Z} \end{aligned}$$

Como mencionado no início desta seção, o problema resultante $\mathbf{LD}_\lambda$, também chamado de *problema lagrangeano*, deve ser fácil de resolver. Normalmente são classificados nesta

categoria problemas resolvidos em tempo polinomial ou pseudo-polinomial. Isso leva a outro ponto estratégico da formulação do problema lagrangeano, que é a escolha do conjunto de restrições a ser relaxado. Um possível mecanismo para guiar esta escolha é *propriedade de integralidade*. Um problema relaxado possui essa propriedade se sua solução se mantém inalterada quando as restrições de integralidade são trocadas pelas respectivas relaxações lineares. No caso de problemas lagrangeanos que possuam a propriedade de integralidade o maior limite inferior (problemas de minimização) atingível é o valor ótimo da relaxação linear do problema original. Para problemas que não possuem essa propriedade, o limite inferior pode atingir valores maiores ou iguais aos da relaxação linear. Uma discussão detalhada destes tópicos pode ser encontrada em [13].

Falta ainda mostrar como o problema relaxado gera um limite inferior para o problema original. O primeiro fato a ser notado é que toda solução viável de P também é viável para LD_λ. Em especial, a solução ótima de P, denotada por y^*, é solução viável para LD_λ. Logo $Z_D(\lambda) \leq cy^* + \lambda(b - Ay^*)$ e como $Z = cy^*$ podemos reescrever a equação acima da seguinte forma:

$$Z_D(\lambda) - \lambda(b - Ay^*) \leq Z \tag{2.4}$$

Agora basta notar que $\lambda(b - Ay^*) \leq 0$ para concluir que $Z_D(\lambda) \leq Z$.

Dado um vetor $\lambda \geq 0$ viu-se que $Z_D(\lambda)$ é um limite inferior para o problema original, no entanto este valor pode estar tão longe do ótimo que não seja de nenhuma utilidade prática. Na verdade, o que se quer achar são os valores dos multiplicadores que resultem no maior limite inferior possível. Isso é o mesmo que resolver o problema abaixo, o qual é denotado por LD.

$$\begin{aligned} Z_D &= \max Z_D(\lambda) \\ &s.a. \quad \lambda \geq 0 \end{aligned}$$

O problema enunciado acima é conhecido como Problema Lagrangeano Dual. Existem duas técnicas principais para sua resolução. A primeira é conhecida como ajuste de multiplicadores, que é explicada em detalhes em [14]. A segunda, utilizada neste trabalho, é a otimização por subgradiente, que será discutida a seguir.

2.2.2 Otimização por Subgradiente

Suponha que o conjunto de soluções viáveis de $\mathbf{LD}_\lambda$, denotado por Y, é finito. Então, Y pode ser representado como $Y = \{y^t \mid t = 1 \ldots T\}$. Essa nova representação permite criar um problema $\overline{LD}$, equivalente a LD, como se segue.

$$\begin{aligned} Z_D &= \max w \\ s.a.\ & w \leq cy^t + \lambda(b - Ay^t), t = 1 \ldots T \end{aligned}$$

Como pode ser visto acima, o lado direito de cada restrição de $\overline{LD}$ é uma função linear de λ. Isso mostra que $Z_D(\lambda)$ é o espaço limitado abaixo de um conjunto finito de funções lineares. Essas características implicam que $Z_D(\lambda)$ tem algumas propriedades que a tornariam simples de resolver por um método de *hill climbing*. Essas propriedades são continuidade e concavidade. Contudo, é fácil ver que $Z_D(\lambda)$ é uma função linear por partes e por isso, não diferenciável em todos os pontos. Isso faz com que não se possa utilizar o gradiente como direção de subida, entretanto como será visto adiante, $Z_D(\lambda)$ é sub-diferenciável em todos os pontos e, deste modo, é possível trocar o gradiente pelo subgradiente.

Um vetor φ é chamado de subgradiente de $Z_D(\lambda)$ no ponto $\overline{\lambda}$ se satisfaz a seguinte desigualdade:

$$Z_D(\lambda) \leq Z_D(\overline{\lambda}) + \varphi(\lambda - \overline{\lambda}) \quad \forall\lambda \tag{2.5}$$

É fácil ver que, se y' resolve $LD_{\overline{\lambda}}$ então $\varphi = (b - Ay')^T$ é subgradiente de $Z_D(\lambda)$ em $\overline{\lambda}$. Basta substituir φ em 2.5 e obtem-se:

$$Z_D(\lambda) \leq cy' + \lambda(b - Ay')$$

O que é válido para toda solução do problema dual, como pode ser visto na formulação de $\overline{LD}$.

Pode-se então obter um sequência de valores $\{\lambda^k\}$, dado um λ^0, apenas adaptando método do gradiente substituindo-se a direção de busca por aquela do vetor subgradiente. A sequência é gerada com a seguinte fórmula:

$$\lambda^{k+1} = \lambda^k + \theta_k(b - Ay^k) \tag{2.6}$$

onde y^k é a solução ótima de LD_{λ^k} e θ_k é um escalar positivo chamado de *tamanho de passo.*

Existe um resultado teórico que mostra que, quando $k \to \infty$, $Z_D(\lambda^k) \to Z_D$ se $\theta_k \to 0$ e $\sum_{i=0}^{k} \theta_i \to \infty$. A primeira condição garante que, conforme se chega próximo da solução do problema, o tamanho de passo diminui, ou seja, uma vez perto do ótimo não se deseja sair. A segunda garante que não se pare prematuramente, ou seja, antes de se chegar ao ótimo. O tamanho de passo normalmente usado é dado pela fórmula que se segue:

$$\theta_k = \frac{\alpha_k(\xi Z_{HP} - Z_D(\lambda^k))}{\|b - Ay^k\|^2} \tag{2.7}$$

onde α_k é um escalar que satisfaz $0 < \alpha_k \leq \alpha_0$, Z_{HP} é um limite superior para o valor de Z_D e ξ é um valor utilizado para que o tamanho de passo não fique muito pequeno conforme $Z_D(\lambda_k)$ se aproxima de Z_{HP}.

A sequência α_k tem também sua própria fórmula de atualização. O valor inicial é $\alpha_0 = 2$ e α_k é dividido por 2 sempre que $Z_D(\lambda^k)$ falha em achar um valor melhor para Z_D após um certo número pré-determinado de iterações. Este último parâmetro é denotado aqui por γ.

2.2.3 Heurística Lagrangeana e Redução do Problema

No final da seção anterior foi mencionado o valor Z_{HP}, que é um limite superior para o valor da solução ótima do problema. O modo mais comum de se obter um valor para Z_{HP} é através do uso de uma heurística lagrangeana.

Uma heurística lagrangeana é um algoritmo que utiliza os resultados obtidos durante a resolução do problema lagrangeano dual, para obter soluções viáveis do problema original. Existem basicamente dois resultados da relaxação lagrangeana que podem ser usados. A solução do problema lagrangeano e os custos calculados durante a relaxação. Como mencionado em [14], algoritmos simples, muitas vezes gulosos, são os que obtem os melhores resultados e podem ser construídos utilizando os dados acima.

Como visto na seção anterior, a solução do problema lagrangeano dual é feita iterativamente utilizando o método do subgradiente. Da mesma forma, a heurística pode ser executada toda vez que se resolve a relaxação do problema para um determinado vetor λ.

O método que compreende a resolução da relaxação lagrangeana acrescido da heurística será chamado de procedimento lagrangeano. Este método ainda pode ser incrementado utilizando técnica de redução do problema (fixação de variáveis) e geração de cortes, como será visto mais a frente. Algumas vezes durante o texto o procedimento lagrangeano será simplesmente chamado de heurística, uma vez que este não garante a obtenção da solução ótima.

A redução do problema conseguida através da fixação de variáveis tem o objetivo principal de diminuir o tempo de resolução de cada iteração do método. Nesse sentido, o tempo extra gasto no processo de fixação deve ser tal que seja compensado pelo ganho que estará trazendo nas iterações seguintes.

O método de fixação de variáveis é bastante simples, basta incluir uma restrição do tipo $y_k = 1$ ou $y_k = 0$ no problema relaxado para um determinado índice k. Obtem-se então a solução deste novo problema e se este valor for maior que Z_{HP}, a variável pode ser fixada no valor contrário ao do lado direito da restrição que foi incluída. A idéia do método é que resolvendo a relaxação com a restrição adicional obtem-se um limite inferior para o novo problema. Se este limite é maior que o valor de uma solução viável do problema original, claramente a variável não pode assumir o valor proposto pela restrição restando, portanto,

apenas um valor possível.

2.2.4 Planos de Corte

Nesta seção descreve-se o uso de planos de corte juntamente com relaxação lagrangeana. O algoritmo usando essa técnica é chamado na literatura de *Relax & Cut* [15, 17].

A teoria dos métodos de planos de corte foi introduzido na década de 50 por Gomory para resolver problemas de programação linear inteira (PLI). A idéia básica deste algoritmo é obter a solução para o problema de relaxação linear e, enquanto essa não for inteira, eliminá-la acrescentando novas inequações válidas (planos de corte) para o problema, que eliminem soluções contínuas sem contudo eliminar soluções viáveis inteiras. Para uma discussão mais aprofundada recomenda-se a leitura de [23] e [24].

O método Relax & cut é uma adaptação quase natural deste método descrito acima. Neste algoritmo, para um dado λ gerado pela sequência do método subgradiente, obtem-se um vetor solução inteiro y^* associado. O próximo passo é encontrar desigualdades do problema original P que sejam violadas por y^*. Finalmente, para que não haja alteração do problema lagrangeano, essa desigualdade é dualizada (relaxada) e adicionada à formulação do problema.

Existem duas diferenças básicas entre os dois métodos. No método original, que resolve a relaxação linear do problema, a adição de cortes implica que a solução corrente y^* jamais será encontrada novamente e que o valor da solução na próxima iteração será sempre melhor ou igual ao valor atual. O mesmo não se verifica no Relax & Cut, uma vez que o corte inserido é dualizado e, portanto, o conjunto solução do problema relaxado permanece inalterado.

Então, por que utilizar cortes com relaxação lagrangeana? O primeiro motivo é que, conforme o método do subgradiente avança, as soluções obtidas tendem a se aproximar da solução da relaxação linear do problema. Logo um corte gerado para aquela solução também pode cortar soluções fracionárias da relaxação linear, possibilitando assim o aumento do limite da relaxação lagrangeana. Um segundo motivo, mais prático do que teórico, é que a inserção de cortes penalizando as infactibilidades pode fazer com que o método do subgradiente tenha convergência mais rápida.

Um problema que surge neste método é que o número de cortes inseridos ao longo do algoritmo pode ser excessivamente grande, prejudicando o seu desempenho geral. Além disso, como esses cortes são gerados para soluções que podem não ter relação com a solução do problema de relaxação linear, estes podem se tornar inúteis ($\lambda = 0$) em iterações posteriores. Isso exige algum mecanismo de gerência que permita, não só a adição, mas também a remoção dos cortes da formulação.

Uma discussão sobre o mecanismo citado acima, bem como outros tópicos de implemen-

tação podem ser encontrados na seção 5.5.

2.3 Conjuntos Independentes

Nesta seção será apresentado o problema dos *conjuntos independentes* de um grafo. A razão para isto é lançar as bases teóricas que permitirão a obtenção de desigualdades válidas para o **RG-NLP**.

Seja $G = (V, E)$ um grafo não orientado, onde V representa o conjunto de vértices e E é o conjunto de arestas. Um conjunto independente de G é um subconjunto $V' \subseteq V$ tal que para todo par de vértices $\{v_i, v_j\}$ em V' a aresta (v_i, v_j) não está em E. Se forem atribuidos custos c_i a cada vértice v_i, o problema de maximização da soma destes custos é conhecido como *problema do conjunto independente de custo máximo em G*. Esse é um problema clássico de teoria dos grafos e é provado ser NP-difícil.

Esse problema foi intensamente estudado utilizando-se técnicas de PLI. Um possível modelo é apresentado a seguir. Se forem associadas variáveis binárias y_i de custo c_i a cada vértice v_i, de modo que y_j assuma o valor 1 se e somente se o vértice v_j esteja no conjunto independente, então pode-se descrever a restrição acima da seguinte forma:

$$y_i + y_j \leq 1, \ \forall (v_i, v_j) \in E$$

A desigualdade descrita acima garante que se dois vértices de V possuem uma aresta então eles não podem fazer parte do mesmo conjunto independente. Por este motivo esta é conhecida como desigualdade de aresta.

O problema enunciado acima pode então ser formulado da seguinte maneira:

$$\begin{aligned} Z \quad = \quad & \max \sum_{v_i \in V} c_i y_i \\ s.a. \quad & \\ & y_i + y_j \leq 1 \quad \forall\, (v_i, v_j) \in E \\ & y_i \in \{0,1\} \quad \forall\, v_i \in V \end{aligned}$$

O problema do conjunto independente é um caso particular do problema de empacotamento (*set packing*), que é outro problema clássico de otimização combinatória. Esse problema é caracterizado por possuir restrições de menor ou igual, cujo lado direito tem valor 1 e cujas variáveis também possuem coeficiente 0 ou 1. Ou seja, na forma matricial o problema pode ser descrito por $\{\max cy \mid Ay \leq 1, y \in \{0,1\}^n\}$, onde $a_{ij} \in \{0,1\}$.

Por causa do grande número de aplicações que este problema possui, a envoltória convexa de suas soluções viáveis foi intensamente estudada e classes de desigualdades válidas foram

encontradas, algumas delas definindo facetas deste politopo. Deve-se observar que toda desigualdade válida para um problema de empacotamento também é válida para o problema de partição correspondente da forma $\{\max cy \mid Ay = 1, y \in \{0,1\}^n\}$, onde $a_{ij} \in \{0,1\}$. Sendo assim, a próxima seção apresentará algumas desigualdades válidas para o problema de empacotamento, que serão utilizadas neste trabalho, uma vez que, como será visto no capítulo 3, o **RG-NLP** pode ser formulado como um problema de partição de conjuntos.

2.3.1 Desigualdades de Clique

A desigualdade de clique é uma generalização da desigualdade de aresta para mais de dois nós. Neste caso, determina-se um subconjunto $V' \subseteq V$ tal que no máximo um dos vértices pode fazer parte de um conjunto independente. Para isso, V' tem que ser tal que todos os seus vértices sejam conectados dois a dois, ou seja, V' é um subgrafo completo ou uma clique de G. O grafo da figura 2.1 (a) traz um exemplo de um clique de tamanho 3. Abaixo segue um exemplo de desigualdades que podem gerar o grafo citado.

$$\begin{array}{llll} y_1 + & y_2 + & \ldots & \leq 1 \\ y_1 + & y_3 + & \ldots & \leq 1 \\ y_2 + & y_3 + & \ldots & \leq 1 \end{array}$$

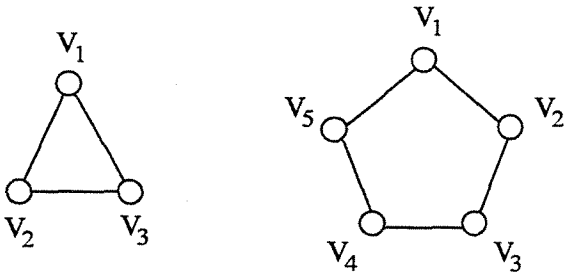
Neste exemplo é possível ver que y_1 e y_2 não fazem parte de uma mesma solução devido a primeira desigualdade. O mesmo vale para os outros pares de variáveis representadas na segunda e na terceira desigualdades. Esse raciocínio leva à conclusão de que apenas uma dessas variáveis pode fazer parte de uma solução inteira que não viole essas restriçoes. Com isso, é possível inserir uma nova restrição no modelo que seja válida para poliedro inteiro. A restrição de clique neste caso é representada pela desigualdade $y_1 + y_2 + y_3 \leq 1$.

O raciocínio utilizado acima pode ser estendido para obter a desigualdade de clique genérica que se segue.

$$\sum_{v_i \in V'} y_i \leq 1$$

É fácil ver que para o caso em que $|V'| = 2$, a desigualdade encontrada se resume a uma desigualdade de aresta.

Uma clique em G formada pelos vértices em V' é dita maximal se não existe V'' que defina um clique tal que $V' \subset V''$. As cliques maximais tem a importante propriedade de definerem facetas para o politopo dos conjuntos independentes [19].



(a) (b)

Figura 2.1: Exemplos de grafos suporte para geração de desigualdades de clique e ciclo ímpar.

2.3.2 Desigualdades de Ciclo Ímpar

Antes da definição de ciclo ímpar são necessárias algumas outras definições. Suponha que $V' = \{v_1, v_2, \ldots, v_n\}$ seja um sequência de vértices em G. O conjunto V' define um caminho se $(v_i, v_{i+1}) \in E$ para $1 \leq i \leq n-1$. Este caminho forma um ciclo se $v_1 = v_n$. O ciclo forma um *buraco* se este não possui cordas. Uma aresta (v_i, v_j) é uma corda do ciclo formado por V' se $j \neq i+1$. Finalmente, se $|V'| = 2k+1$, onde $k \in \mathbb{N}^+$, então tem-se um caminho, ciclo ou buraco ímpar. A desigualdade apresentada a seguir é denominada em inglês como *odd hole*, que poderia ser melhor traduzido para *buraco ímpar*. Apesar disso, na literatura técnica em português usa-se o termo ciclo ímpar.

Usando raciocínio semelhante ao desenvolvido na seção anterior para as cliques, o conjunto de desigualdades abaixo permite a construção do grafo da figura 2.1 (b).

$$
\begin{array}{lllll}
y_1 + & y_2 + & & & \ldots \leq 1 \\
 & y_2 + & y_3 + & & \ldots \leq 1 \\
 & & y_3 + & y_4 + & \ldots \leq 1 \\
 & & & y_4 + \quad y_5 & \ldots \leq 1 \\
y_1 + & & & \quad y_5 & \ldots \leq 1
\end{array}
$$

Pode-se então adicionar a desigualdade $y_1 + y_2 + y_3 + y_4 + y_5 \leq 2$ ao modelo. Essa desigualdade mostra que no máximo duas dessas variáveis podem pertencer a esse conjunto independente. Este é um fato fácil de constatar pois, já que cada dois vértices consecutivos no ciclo estão ligados por uma aresta, no máximo um deles pertence ao conjunto independente. Desta forma, escolhendo-se um vértice arbitrário como ponto de partida, pode-se percorrer o ciclo colocando os vértices visitados nas posições ímpares no conjunto indepen-

dente. Contudo, o último vértice visitado é vizinho do primeiro e, portanto não poderá pertencer ao conjunto independente que ficará então com 2 vértices.

O raciocínio utilizado acima pode ser extendido para obter a desigualdade de ciclo ímpar genérica que se segue:

$$\sum_{v_i \in V'} y_i \leq k$$

Um fato interessante de se notar é que quando $|V'| = 3$ a desigualdade de ciclo ímpar é igual a de clique. Apesar disto esta não define uma faceta em todos os casos. Um procedimento de *lifting* para se obter uma faceta a partir de um ciclo ímpar sem cordas pode ser encontrado em [18, 19]. No entanto esse procedimento parece ser de pouca utilidade prática, uma vez que, para ser levado a termo, é necessária a resolução de várias instâncias menores do problema dos conjuntos independentes de peso máximo.

Capítulo 3

Formulação do Problema RG-P

Neste capítulo será apresentada uma formulação de programação linear inteira 0-1 para o **RG-P**, como descrito no capítulo 1. Antes da formulação serão apresentadas algumas definições e resultados básicos que serão essenciais durante todo o transcorrer deste texto. Grande parte destas definições são baseadas no trabalho descrito em [9] visando, assim, manter uma notação comum.

3.1 Definições e Resultados Básicos

Esta seção lista algumas definições e teoremas básicos que serão úteis para formular o **RG-P** como um problema de partição de conjuntos. A definição que se segue nos permite enunciar o teorema de Lingas *et al* [1], que caracteriza os segmentos presentes numa solução ótima do **RG-P**.

Definição 3 *Seja $I = (R, P)$ uma instância do **RG-P**, a grade induzida pelos pontos de P, denotada por **GI(P)**, é o conjunto finito de segmentos de reta definidos no interior de R pelas retas verticais e horizontais que passam pelos pontos de P.*

A figura 3.1 mostra uma instância do **RG-P** e sua respectiva grade induzida.

Teorema 1 *[1] Se um segmento está numa solução ótima, então ele pertence a **GI(P)**.*

A interseção das retas de **GI(P)** compõe o conjunto de pontos da grade induzida, que é denotado por **GP(P)**. Esse conjunto pode ser dividido em dois. O primeiro é o conjunto dos *pontos terminais*, isto é, os pontos que pertencem a P, e o segundo é o conjuntos dos *pontos de Steiner* denotado por **SP(P)**, que são os restantes. Na figura 3.1 os pontos cheios representam os pontos terminais e os brancos representam os pontos de Steiner.

Agora serão apresentadas algumas configurações que permitirão descrever as soluções viáveis do **RG-P**.

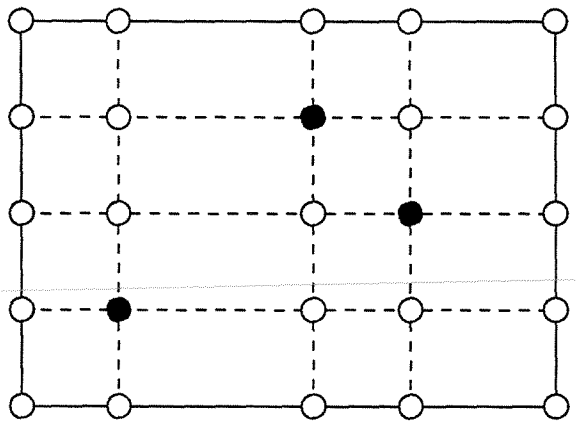


Figura 3.1: Uma instância $I = (R, P)$ do **RG-P**. Os segmentos pontilhados representam **GI(P)**, os pontos cheios são terminais e os brancos são pontos de Steiner.

Definição 4 *Seja um ponto $p \in$ **GP(P)** e S um subconjunto de segmentos tal que $S \subseteq$ **GI(P)** . Define-se o grau de p relativo a S, cuja notação é $g(p)$, como sendo o número de segmentos de S que incidem em p.*

Definição 5 *[9] Seja $I = (R, P)$ uma instância do **RG-P** e $S \subseteq$ **GI(P)** um conjunto de segmentos. Um ponto $p \in GP(P)$ define um joelho se $g(i) = 2$ e os segmentos que incidem em p são ortogonais. Se $g(p) = 1$, então temos uma configuração denominada ilha.*

O teorema a seguir mostra as propriedades geométricas mínimas para se obter um solução viável para o problema, de acordo com a definição do problema enunciada nos capítulos anteriores.

Teorema 2 *[9] Seja $I = (R, P)$ uma instância do **RG-P**. Se S é um conjunto de segmentos de **GI(P)** que não contém joelhos nem ilhas e S é tal que em todo ponto de P incidem pelo menos dois segmentos de S, então S é solução factível de I.*

Como será visto mais adiante vamos modelar o **RG-P** como um problema de partição de conjuntos *(Set Partitioning Problem)*. Desta maneira, a resolução do **RG-P** pode ser entendida como a partição do retângulo R em retângulos menores de acordo com as definições do problema no capítulo 1. Para chegar a formulação ainda são necessárias algumas definições sobre os retângulos que são encontrados na grade induzida.

Definição 6 *Seja $I = (R, P)$ uma instância do **RG-P**, e **GI(P)** sua grade induzida. Dizemos que um retângulo é induzido por **GI(P)** se seus vértices estão em **GI(P)**.*

Definição 7 *Um retângulo induzido por* ***GI(P)*** *é dito ser viável se não contêm nenhum ponto de P em seu interior.*

Definição 8 *Um retângulo induzido por* ***GI(P)*** *é dito ser canônico se não contém nenhum outro retângulo induzido por* ***GI(P)***.

Observando as definições acima é fácil ver que todo retângulo canônico é um retângulo viável.

3.2 Uma Formulação de PLI para o RG-P

Esta seção inicia relembrando a definição do problema de partição de conjuntos e apresentando uma formulação de PLI clássica para a sua resolução. Em seguida é mostrado como o **RG-P** pode ser classificado desta maneira.

Considere um conjunto $H = \{H_i | i \in I\}$, onde $I = \{1 \ldots m\}$. Considere ainda um conjunto de subconjuntos de H dado por $K = \{K_j | j \in J\}$, onde $J = \{1 \ldots n\}$. Desta forma, um conjunto $J' \subseteq J$ define um partição de H se $\bigcup_{j \in J'} K_j = H$ e $K_j \bigcap K_l = \emptyset$, para todo par de índices $j, l \in J'$ com $j \neq l$.

Se forem associados aos elementos de K custos $\{c_j | j \in J\}$, então o custo da partição induzida por J' é computado como $\sum_{j \in J'} c_j$. Para resolver o problema de partição de conjuntos deve-se achar a partição de menor custo.

Agora que a definição do problema está clara, é necessário apresentar a formulação de PLI para a sua resolução. Então, para todo $i \in I$, seja J_i o conjunto de índices $j \in J$ tal que H_i está contido K_j. Analogamente, para todo $j \in J$, seja $I_j \subseteq I$, o conjunto de índices dos elementos de H que pertencem a K_j. Por fim, para todo $j \in J$, define-se a variável y_j que controla a inclusão ou não de K_j na partição. Então, a formulação para o problema de partição de conjuntos é dada como se segue.

$$\min \{ \sum_{j \in J} c_j y_j \mid y \in \Re_0 \bigcap \mathbb{Z}^{|J|} \} \tag{3.1}$$

onde $\Re_0$ é a região poliédrica descrita por:

$$\sum_{j \in J_i} y_j = 1 \quad \forall\, i \in I \tag{3.2}$$

$$0 \leq y_j \leq 1 \quad j \in J \tag{3.3}$$

Agora para modelar o problema **RG-P** como um problema de partição de conjuntos, deve-se definir os conjuntos H e K, bem como atribuir apropriadamente custos c_j para cada elemento $K_j \in K$. Essas definições seguem na lista abaixo.

- H é o conjunto de todos os retângulos canônicos em **GI(P)** (Vide definição 8).
- K é o conjunto de todos os retângulos viáveis (Vide definição 7).
- O custo c_j associado a cada $K_j \in K$ é definido como o perímetro ponderado do retângulo K_j. Esse perímetro ponderado é calculado da seguinte maneira: Se R_j tem um lado l que está na borda do retângulo R, então, l tem peso igual a 2, senão tem peso 1.

Na figura 3.2 pode-se observar que a união de todos os retângulos canônicos resulta no retângulo R e mais ainda, que todos os retângulos viáveis podem ser construídos através da união de retângulos canônicos adjacentes. Em outras palavras, queremos que cada retângulo canônico seja coberto por um e somente um retângulo viável. Isto é o mesmo que dizer que R está sendo particionado em retângulos viáveis.

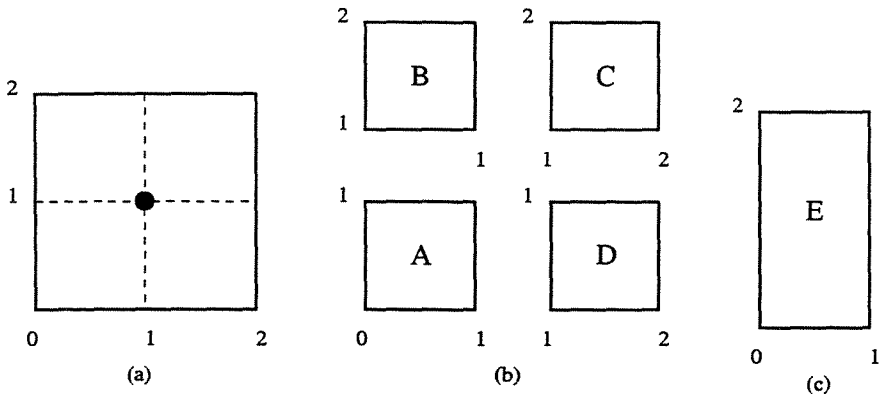


Figura 3.2: (a) Uma Instância do **RG-P**. (b) Retângulos canônicos em **GI(P)**. (c) Um retângulo viável.

Os argumentos acima mostram que essa formulação leva a uma solução viável do problema conforme definido no capítulo 1. Falta ainda explicar como a soma dos custos propostos acima leva ao comprimento total dos segmentos de reta internos a R de custo mínimo. O teorema que se segue mostra que o valor da solução obtida pela formulação acima se relaciona linearmente com a soma dos segmentos internos a R.

Teorema 3 *[9] Seja $I = (R, P)$ uma instância do **RG-P**. Seja y^* uma solução ótima inteira de valor Z^*, e seja S^* a soma dos segmentos internos de R definidos por y^*. Então, $Z^* = 2S^* + 2Per(R)$, onde $Per(R)$ é o perímetro do retângulo R.*

Corolário 1 *Z^* é par.*

Outro fato interessante de se notar é que há uma restrição de igualdade para cada retângulo canônico e uma variável para cada retângulo viável. Isso resulta em $O(|P|^2)$ restrições e $O(|P|^4)$ variáveis, onde $|P|$ é a cardinalidade do conjunto de pontos terminais da instância em questão.

Capítulo 4

Reduções Geométricas e Novas Desigualdades Válidas

Neste capítulo serão descritas e provadas algumas propriedades geométricas do **RG-NLP**. A partir destas propriedades serão abordadas técnicas para reduzir a formulação original do problema. Será provado que dois conjuntos de variáveis e um conjunto de restrições podem ser eliminados, a priori, sem que se perca uma solução ótima do problema. Além disso, algumas dessas propriedades serão usadas para a obtenção de desigualdades válidas para o problema.

O próximo teorema mostra que algumas configurações, apesar de fazerem parte de soluções viáveis, podem ser descartadas, uma vez que a partir destas obtemos soluções viáveis de menor custo. Esse teorema é a base para as outras propriedades que serão mostradas no decorrer do capítulo.

Teorema 4 *Numa solução ótima do* ***RG-NLP****, se* $p \in P$ *então* $g(p) = 2$*. Se* $p \in SP(P)$ *então* $g(p) \in \{0, 2, 3\}$*.*

Prova

- *Se* $p \in P \Rightarrow g(p) = 2$*. Suponha que a configuração (a) da figura 4.1 faça parte de uma solução ótima. De acordo com a situação ilustrada, vê-se que o grau do ponto terminal p é 4. Além disso, por ser uma solução ótima, o número de segmentos horizontais incidentes no segmento* $\overline{p'_1 p}$ *pelo lado direito deve ser igual ao número de segmentos que chegam pela esquerda. Caso contrário, o valor da solução ótima seria diminuído arrastando* $\overline{p'_1 p}$ *no sentido que possui mais segmentos. Agora, arrastando* $\overline{p'_1 p}$ *para a direita, a solução não terá seu valor alterado e ainda será viável. Além do mais, essa operação fará com que o grau de p se reduza de uma unidade. Podemos continuar deslocando este segmento para a direita até que ele encoste em outro segmento da*

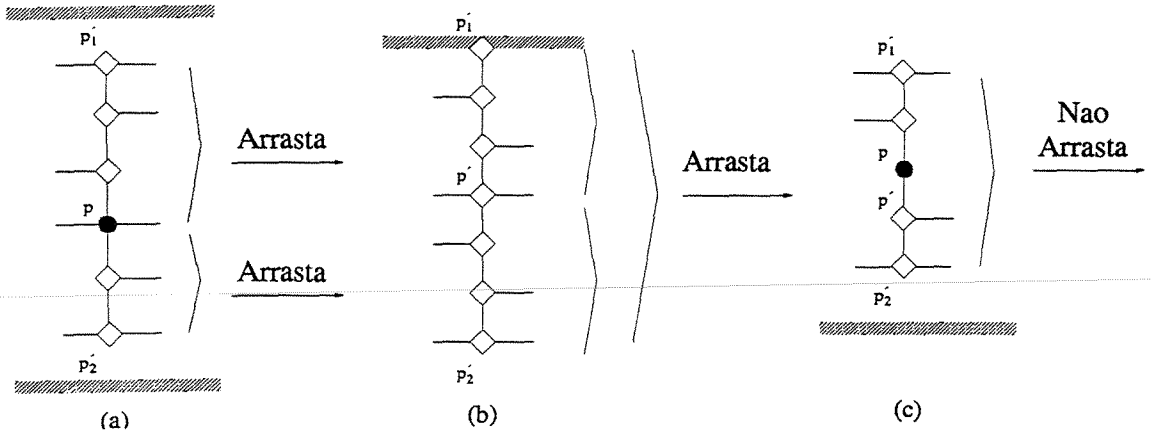


Figura 4.1: Possibilidades de movimento de segmentos de soluções viáveis visando diminuir o valor da solução. Os círculos representam os pontos terminais e os losangos, os pontos de Steiner.

solução ótima (eventualmente a borda externa do retângulo). Desta maneira, pode-se retirar da solução, sem perda de viabilidade, a interseção entre o segmento arrastado e o atingido diminuindo o custo da solução. O mesmo procedimento pode ser feito para o segmento $\overline{pp_2'}$ reduzindo o grau de p para 2, o que contradiz a suposição feita anteriormente de que a solução inicial era ótima. O mesmo raciocínio poderia ser aplicado se o segmento da figura fosse horizontal.

- *Se $p \in \boldsymbol{SP(P)} \Rightarrow g(p) \in \{0, 2, 3\}$. Pelo teorema 2 sabemos que nenhum ponto pode ter grau 1. Sendo assim, basta provar que nenhum ponto de Steiner pode ter grau 4. Supomos que a configuração (b) da figura 4.1 faz parte de uma solução ótima. Os mesmos argumentos utilizados acima permitem que o segmento $\overline{p_1'p_2'}$ seja arrastado diminuindo o valor da solução o que contradiz a hipótese de que ela é ótima e, portanto, p' não pode ter grau 4.* □

Nas seções seguintes são mostradas algumas aplicações do teorema 4 que permitem reduzir as formulações de PLI para o **RG-NLP** .

4.1 Retângulos com Ponto Terminal no Vértice

A primeira redução diminui o número de variáveis na formulação do problema e pode ser enunciada como se segue.

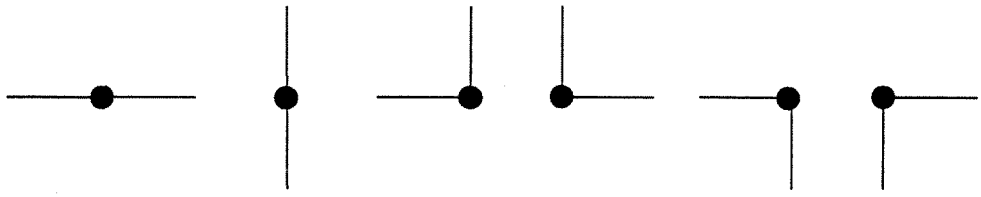


Figura 4.2: Seis modos de um ponto ter grau 2. Em dois deles os segmentos incidentes pertencem à mesma reta e nos quatro restantes, esses segmentos são perpendiculares.

Teorema 5 *Seja y_j a variável associada ao j-ésimo retângulo viável (R_j). Se R_j possui um vértice que é ponto terminal, então em toda solução ótima para o problema tem-se que* $y_j = 0$.

Prova

Seja p o vértice de R_j que é ponto terminal. Pelo teorema 4 sabemos que, em qualquer solução ótima, $g(p) = 2$. Sabemos também pelo teorema 2 que uma solução viável não possui joelhos. Na figura 4.2 vemos que a única maneira de satisfazer ambas as condições é impor que os dois segmentos incidentes em p pertençam a mesma reta. Desta forma R_j não pode fazer parte da solução ótima e, portanto, $y_j = 0$. □

Desta forma, o teorema 5 permite a eliminação de toda variável correspondente a um retângulo viável com vértice num ponto terminal.

4.2 Retângulos com Interseção nos Lados Estendidos

A segunda redução apresentada também diminui o número de variáveis na formulação do problema. No entanto, para apresentá-la é necessário mostrar algumas propriedades adicionais de soluções ótimas do **RG-NLP**.

A configuração (c) da figura 4.1, é construída de tal forma que não podem ser arrastados segmentos, como feito para provar o teorema 4. Nesta configuração chamamos os pontos de Steiner de grau 3, p'_1 e p'_2, de identificadores do ponto terminal p. Formalmente, pode-se definir os pontos identificadores da forma como se segue. Dada uma solução ótima $S \subseteq \mathbf{GI(P)}$ de uma instância do **RG-NLP**, seja $p \in P$ e, sem perda de generalidade, suponha que os dois únicos segmentos incidentes em p mencionados no teorema 4 sejam verticais. Partindo-se de p e percorrendo-se a reta r que contém esses dois segmentos em um sentido arbitrário, seja q o primeiro ponto de Steiner a ser visitado sobre o qual incidem dois segmentos horizontais. O ponto q é dito ser um identificador de p. O outro identificador

pode ser obtido de forma análoga percorrendo-se r a partir de p, mas no sentido inverso. Vale observar que, pelo teorema 4, os identificadores são sempre pontos de Steiner de grau 3. Note que, de acordo com a definição anterior, embora p' tenha grau 3, ele não é identificador de p, mas de outro ponto não representado na figura 4.1.

Teorema 6 *Seja $I = (R, P)$ uma instância do **RG-NLP**. Em qualquer solução ótima todo ponto de Steiner de grau 3 é identificador de um ponto terminal.*

Prova

Seja S um solução ótima. Iniciando num ponto de Steiner de grau 3, considere o ponto adjacente na direção (horizontal ou vertical) que possui um único segmento, se o ponto adjacente for um Steiner de grau 2, ou Steiner de grau 3 com dois segmentos no sentido que está sendo percorrido, então considera-se o próximo ponto adjacente nesta mesma direção. Este procedimento é repetido até que se atinja um ponto terminal, o que prova o teorema, ou até que se atinja um outro ponto de Steiner de grau 3 com dois segmentos perpendiculares à direção seguida. Este seria o caso do segmento $p'_1p'_2$ na figura 4.1 (b). E, como visto na prova do teorema 4, este segmento pode ser arrastado levando a uma solução de custo inferior ao custo de S, o que contraria a hipótese. □

Corolário 2 *Em qualquer solução ótima de uma instância $I = (R, P)$ do **RG-NLP** o número de pontos de Steiner de grau 3 é igual a $2|P|$.*

O teorema acima nos sugere enunciar a seguinte definição.

Definição 9 *Seja R_j um retângulo viável para uma instância do **RG-NLP** (vide figura 4.3). Seja S o segmento que representa um lado de R_j. Se S contém um ponto terminal então o lado estendido relativo a S é o próprio segmento. Caso contrário, define-se um lado estendido de R_j como sendo o segmento obtido ao estendermos S ao longo da reta que o contém e no sentido de seu ponto terminal corretilinear, até o primeiro ponto de Steiner posterior a este terminal.*

No exemplo da Figura 4.3, R_j possui os seguintes lados estendidos $\{ \overline{p_{b1}p'_3}, \overline{p_{b2}p'_2}, \overline{p_{b3}p'_2}, \overline{p'_4p'_3} \}$

O seguinte corolário pode ser derivado do teorema 6 e da definição 9.

Corolário 3 *Se um retângulo R_j está numa solução ótima então os lados estendidos de R_j também fazem parte da solução.*

Por fim, encerramos esta seção enunciando o teorema que resultará em mais uma redução da formulação PLI do **RG-P** enunciada no capítulo 3.

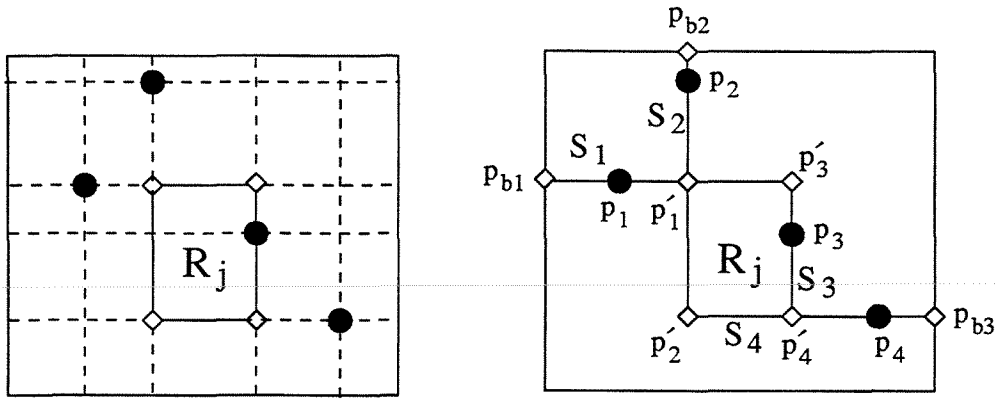


Figura 4.3: (a) Uma instância do **RG-NLP** e suas linhas de grade. (b) Os lados estendidos do retângulo R_j.

Teorema 7 *Seja y_j a variável associada ao j-ésimo retângulo viável $\{R_j\}$. Se dois lados estendidos l_1 e l_2 de R_j se interceptam em um ponto que não é extremo de l_1 nem de l_2, então, para toda solução ótima do problema, $y_j = 0$.*

Prova

Usando como exemplo a figura 4.3, suponha que $y_j = 1$, ou seja, R_j está numa solução ótima S. Como S não pode ter joelhos (Teorema 2) e cada ponto de Steiner tem no máximo grau 3 (Teorema 4), então S pode conter o lado estendido $\overline{p_{b1}p'_3}$ ou $\overline{p_{b2}p'_2}$, mas não ambos. No entanto, qualquer uma das escolhas nos impossibilita de satisfazer o Teorema 6 já que p'_1 não seria identificador de nenhum ponto terminal. Chegamos a uma contradição e, portanto, $y_j = 0$ em qualquer solução ótima. □

4.3 Retângulos Canônicos com Pontos Terminais nos Vértices

O próximo resultado nos permite reduzir o número de restrições na formulação do problema.

Teorema 8 *Seja $p \in P$ um ponto terminal de uma instância do **RG-NLP**. Sejam R_1, R_2, R_3 e R_4, os quatro retângulos canônicos que tem p como vértice. É possível eliminar, sem perda alguma, uma das restrições definidas pelos retângulos canônicos descritos acima.*

Notação: Y_β é uma somatória de variáveis y, onde β é uma sequência de números pertencentes a $\{1,2,3,4\}$ satisfazendo $1 \leq |\beta| \leq 4$, e os índices em β determinam quais

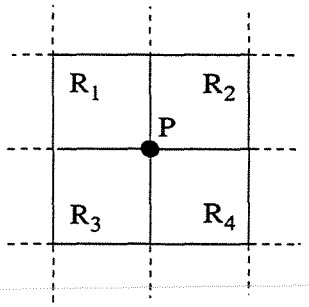


Figura 4.4: Um ponto terminal e seus quatro retângulos canônicos.

retângulos canônicos adjacentes a p que são cobertos pelos retângulos viáveis representados pelas variáveis presentes na somatória.

Prova

Usando a figura 4.4 como exemplo, os valores de algumas destas somatórias de acordo com a notação acima seriam:

- $Y_1 = Y_2 = Y_3 = Y_4 = 0$. *Retângulos com vértice em P podem ser eliminados (Teorema 5).*
- $Y_{12}, Y_{13}, Y_{24}, Y_{34} \geq 0$. *Retângulos viáveis.*
- $Y_{1234} = 0$. *Não existem retângulos viáveis contendo pontos terminais em seu interior.*

Podemos escrever as equações para R_1, R_2, R_3 e R_4 em função dos conjuntos acima descritos.

$$R_1 \ : \ Y_{12} + Y_{13} = 1 \tag{4.1}$$
$$R_2 \ : \ Y_{12} + Y_{24} = 1 \tag{4.2}$$
$$R_3 \ : \ Y_{13} + Y_{34} = 1 \tag{4.3}$$
$$R_4 \ : \ Y_{24} + Y_{34} = 1 \tag{4.4}$$

Note que $(4.2) + (4.3) - (4.1) = (4.4)$. □

A eliminação de dependências lineares no conjunto de restrições conforme sugerido pelo teorema 8 pode tornar mais rápida a resolução do problema de relaxação linear através do método simplex. Apesar disto, o algoritmo descrito neste trabalho não faz uso desta

redução, já que esta foi descoberta muito tardiamente e os testes preliminares feitos com sua aplicação não trouxeram melhora nos tempos de execução. Contudo, não se pode descartar esse como um ponto a ser explorado principalmente porque a redução da matriz traz economia de memória e de tempo de resolução de alguns procedimentos importantes, como será visto no capítulo 5.

4.4 Desigualdades Válidas

Na seção 2.3 foram apresentadas as bases teóricas para obtenção de algumas famílias de desigualdades válidas para o problema de empacotamento (*set packing*) e, conforme visto anteriormente, também válidas para o problema de partição de conjunto (*set partitioning*), que é como modelamos o **RG-P**. Nessa seção será visto como é possível adaptar essas técnicas para encontrar desigualdades válidas para soluções ótimas do **RG-NLP**. Além disso, as propriedades provadas neste capítulo serão utilizadas para mostrar que o número de retângulos em qualquer solução ótima de uma dada instância é constante e, portanto, uma restrição de cardinalidade pode ser adicionada ao nosso modelo.

4.4.1 Cliques e Ciclos Ímpares

Para uma dada instância $I = (R, P)$ do **RG-NLP**, seja $K = \{R_1 \dots R_n\}$ o conjunto de todos os retângulos viáveis de I. Podemos definir o grafo $G = (V, E)$, sendo que $V = \{v_1 \dots v_n\}$ e a cada $R_i \in K$ corresponde um vértice $v_i \in V$, e $E = \{(v_i, v_j) \in V \times V \mid A(R_i \bigcap R_j) \neq \emptyset\}$, onde $A(R_i \bigcap R_j)$ é a área formada pela interseção dos dois retângulos. O grafo G é chamado de *grafo de interseção* de I.

É fácil ver que uma clique maximal obtida neste grafo corresponde a uma desigualdade válida para o problema. Apesar disto, também é fácil verificar que esta desigualdade não corta a solução ótima da Relaxação Linear do problema. Para tanto, basta verificar que esta desigualdade será dominada por uma das restrições do tipo (3.2) definidas para o problema no capítulo 3. O mesmo não foi provado para os ciclos ímpares, no entanto, nos testes experimentais, não foi encontrada nenhuma restrição deste tipo violada.

Os cortes obtidos acima não teriam utilidade na melhora dos limites inferiores da Relaxação Linear e, como este valor é o limite teórico máximo para o valor da Relaxação Lagrangeana, também não seriam úteis nesta relaxação. Sendo assim, é necessário modificar o grafo G descrito acima, e isto pode ser conseguido utilizando o Teorema da Interseção Estendida, que segue após a definição.

Definição 10 *Diz-se que dois retângulos viáveis R_i e R_j têm interseção estendida, denotada por $R_i \wedge R_j$, se existe $p \in \boldsymbol{GP(P)}$ tal que $l_i \bigcap l_j = p$, onde l_i e l_j são dois lados estendidos de R_i e R_j, respectivamente.*

Teorema 9 *Sejam R_i e R_j retângulos viáveis para uma instância do* ***RG-NLP****, y_i e y_j suas variáveis associadas, se $R_i \wedge R_j$, então $y_i + y_j \leq 1$.*

Prova

Vamos supor que $y_i = 1$ e $y_j = 1$. Pelo Corolário 3, isto implica $g(p) = 4$. O que contradiz o teorema 4 □

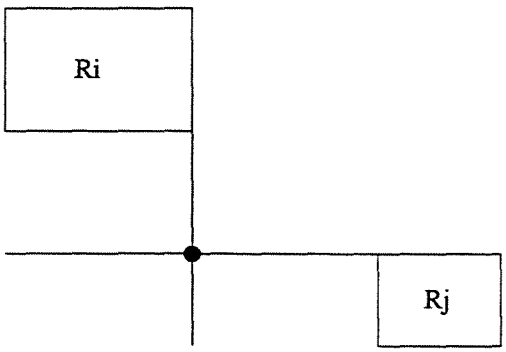


Figura 4.5: Um exemplo onde os retângulos R_i e R_j possuem interseção estendida.

A partir da definição acima, é fácil ver que todo par de retângulos que têm interseção de área também terá interseção estendida.

Desta maneira redefine-se o conjunto de arestas do grafo de interseções de modo que $E' = \{(v_i, v_j) \in V \times V \mid R_i \wedge R_j \neq \emptyset\}$. Os experimentos computacionais realizados neste trabalho mostraram que podem ser obtidas cliques e ciclos ímpares que estão violados para uma solução ótima do problema de relaxação linear. Este novo grafo G' será chamado de *grafo de interseção estendida.*

4.4.2 Restrição de Cardinalidade

O Teorema 6 nos possibilita provar que o número de retângulos em uma solução ótima pode ser calculado a partir do número de pontos terminais da instância. Esse resultado nos leva à obtenção de uma restrição de cardinalidade que será muito útil na construção do problema relaxado, como será visto na seção 5.2.

Teorema 10 *Seja $I = (R, P)$ uma instância do* ***RG-NLP****. O número de retângulos em qualquer solução ótima é $|P| + 1$.*

Prova

Consideremos os seguintes fatos:

- *Todos os vértices de um retângulo viável pertencentes à solução são pontos de Steiner de grau 3 ou um dos 4 vértices de R.*
- *Todo ponto de Steiner de grau 3 é vértice de exatamente 2 retângulos.*
- *Cada um dos 4 vértices de R é também vértice de um retângulo da solução ótima.*
- *A cada ponto terminal correspondem dois identificadores, ou seja, dois pontos de Steiner de grau 3.*

Agora, se r é o número de retângulos na solução ótima e s é o número de pontos de Steiner de grau 3, então:

$$r = \frac{2s+4}{4} = \frac{2 \times 2|P|+4}{4} = |P|+1$$

□

Corolário 4 *A igualdade* $\sum_{j=1}^{n} y_j = |P| + 1$ *é válida para toda solução ótima do* ***RG-NLP****.*

O resultado anterior foi provado de maneira independente por Ferreira, Pina e Fernandes [29] utilizando outros argumentos geométricos.

Capítulo 5

O Algoritmo

Neste capítulo será explicada em detalhes a heurística lagrangeana desenvolvida para a resolução do **RG-NLP**. A primeira seção mostra uma visão geral do algoritmo, as seções seguintes exploram mais detalhadamente cada procedimento e, por fim, são abordadas as principais idéias utilizadas na implementação.

5.1 Uma Visão Geral do Algoritmo

Uma visão geral do algoritmo pode ser encontrada no fluxograma da figura 5.1. Nesta figura estão representados o que entendemos serem os passos básicos para a realização de um algoritmo lagrangeano com inserção de planos de corte (*Relax and Cut*). A idéia desde o início deste trabalho foi desenvolver um conjunto de programas que pudesse ser utilizado como base para resolução de problemas combinatórios através de algoritmos deste tipo, bem como facilitar os diversos testes que seriam feitos especificamente para o **RG-NLP**. No entanto, como o objetivo principal deste trabalho é a resolução deste problema particular tivemos que abrir mão do desenvolvimento de uma ferramenta mais geral. Apesar disto, vale ressaltar que este ambiente básico foi utilizado em outro trabalho no Instituto de Computação da Unicamp [11] e os primeiros resultados para um problema de planejamento de viagens no transporte coletivo foram obtidos em menos de uma semana de programação.

Nesta abordagem caracterizamos cinco procedimentos principais. São eles:

- **Relaxação:** Cálculo dos custos lagrangeanos e da solução do problema relaxado baseado nos multiplicadores válidos na iteração corrente.

- **Heurística:** Procedimento para encontrar uma solução primal viável para o problema.

- **Fixação de variáveis:** Procedimento para verificar, através de cálculos e testes lógicos, se uma variável pode ser fixada em zero ou um.
- **Geração de cortes:** Procedimento para gerar desigualdades válidas para o problema.
- **Subgradiente:** Cálculo dos multiplicadores lagrangeanos para a próxima iteração.

Esses procedimentos são executados ciclicamente de acordo com a ordem do fluxograma da figura 5.1 até que o teste de parada seja satisfeito. Esta última condição acontece quando for atingido um limite máximo de iterações, ou quando a diferença entre o limite superior (resultado da heurística) e o limite inferior (valor da relaxação lagrangeana) seja suficientemente pequeno para que se prove que a solução corrente é ótima.

5.2 O Problema de Relaxação Lagrangeana para o RG-NLP

Vamos agora definir a estrutura do problema relaxado **(PR)**. Para isto utilizaremos a formulação do problema descrita no capítulo 3 e a restrição de cardinalidade do corolário 4.

A restrição de cardinalidade é uma restrição válida para uma solução ótima do **RG-NLP**, logo podemos adicioná-la à formulação original sem alteração do resultado do problema. Essa será a formulação a partir da qual construiremos o problema relaxado.

Obtemos o problema relaxado dualizando as restrições que caracterizam o problema de partição de conjuntos (restrições do tipo (3.2) definidas no capítulo 3), mantendo-se como única restrição a de cardinalidade. Desta maneira, definimos a região poliédrica $\Re_1$ como se segue.

$$\begin{aligned} &\sum_{j=1}^{n} y_j = |P| + 1 \\ &0 \leq y_j \leq 1, \quad j = 1 \ldots n \end{aligned} \tag{5.1}$$

Assim, se λ é o vetor de multiplicadores de Lagrange, podemos escrever o **PR** da seguinte maneira.

$$\min\{\sum_{i=1}^{m}\sum_{j=1}^{n}(c_j - a_{ij}\lambda_i)y_j + \sum_{i=1}^{m}\lambda_i \mid y \in \Re_1\} \tag{5.2}$$

ou ainda,

$$\min\{\sum_{j=1}^{n}\overline{c_j}y_j + \sum_{i=1}^{m}\lambda_i \mid y \in \Re_1\}, \text{onde } \overline{c_j} = \sum_{i=1}^{m}(c_j - a_{ij}\lambda_i) \tag{5.3}$$

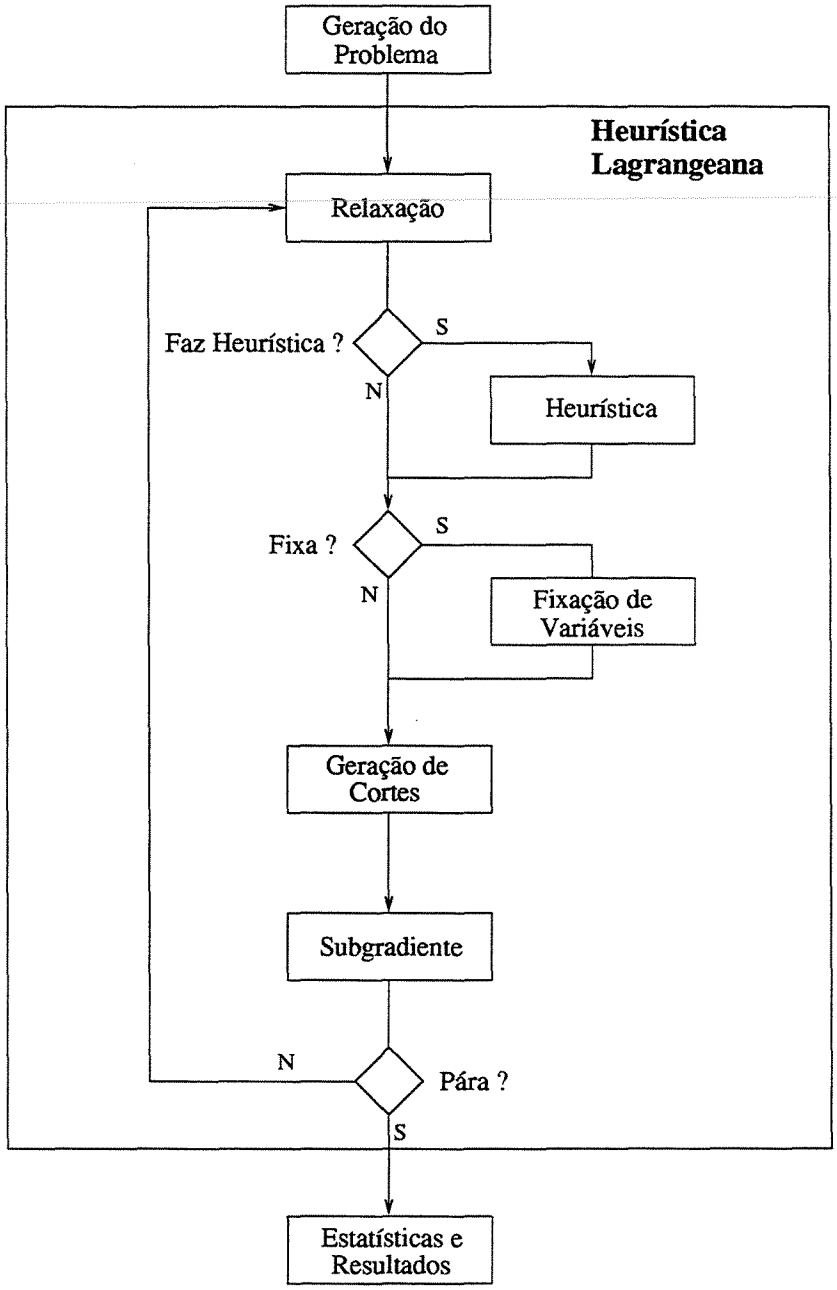


Figura 5.1: Fluxograma de execução da Heurística Lagrangeana com inserção de planos de corte.

UNICAMP

O valor $\overline{c_j}$ é chamado de custo lagrangeano da variável y_j.

A resolução deste problema é bastante simples. Basta encontrar as $|P|+1$ variáveis de menor custo lagrangeano, o que pode ser conseguido em $O(n)$ usando um algoritmo ótimo para resolver o problema de seleção [25].

5.3 O Método do Subgradiente

As bases teóricas do método do subgradiente já foram descritas no capítulo 2, e esta seção terá a função de mostrar os parâmetros específicos que foram utilizados no algoritmo implementado para resolver o **RG-NLP**.

O método do subgradiente é a parte central do procedimento lagrangeano descrito neste trabalho. Desta forma, sempre que for dito que o algoritmo executou mais uma iteração, isso implica que o vetor de multiplicadores de Lagrange teve seus valores alterados.

Para facilitar a explicação dos parâmetros de execução, serão reapresentadas as fórmulas de atualização dos multiplicadores lagrangeanos. Assim, sejam Z_{HP} e Z_R o limite superior e o valor corrente da relaxação lagrangeana, respectivamente. Além disso, seja G o vetor de subgradientes associados às restrições relaxadas. Para a solução corrente $\overline{y}$ de (5.3), G pode ser calculado como se segue.

$$g_i = (1 - a_i\overline{y}),\ i = 1 \ldots m \tag{5.4}$$

O subgradiente é usado para calcular o novo vetor de multiplicadores na iteração K usando a fórmula.

$$\lambda^{K+1} = \lambda^K + \theta^K\, G \tag{5.5}$$

O valor θ^K é chamado de *tamanho de passo* e pode ser calculado da seguinte forma.

$$\theta = \frac{\alpha(\xi\, Z_{HP} - Z_R)}{\sum_{i=1}^{m} g_i^2} \tag{5.6}$$

Onde α é uma constante real que assume valores em $(0, 2]$ e ξ é um valor utilizado para que o tamanho de passo não fique extremamente pequeno quando a diferença entre os limites é bem próxima. Na implementação do algoritmo que fizemos, ξ foi fixado em 1.02. O valor α também tem uma fórmula de atualização segundo a qual ele é dividido à metade toda vez que γ iterações se passam sem que haja melhora em Z_R. O valor escolhido para γ é 80, no entanto, algumas outras discussões sobre essa escolha podem ser encontradas no capítulo 6. Finalmente, o número máximo de iterações escolhido foi 2000. O valor de α e sua fórmula de atualização, bem como o valor de ξ são baseados nos valores sugeridos em [14].

5.4 A Heurística Primal

A heurística de obtenção de uma solução primal viável é um algoritmo essencialmente guloso. Primeiro ordena-se o conjunto de variáveis segundo critérios que serão explicados mais adiante, a seguir este conjunto é percorrido tentando adicionar cada variável ao conjunto solução, desde que o retângulo viável associado à esta variável não tenha interseção estendida com os retângulos associados às variáveis escolhidas até o momento (Teorema 9). O algoritmo obtém uma solução viável sempre que consegue satisfazer a todas as restrições originais do problema, e falha caso chegue ao final do conjunto de variáveis sem cumprir este objetivo. Abaixo segue uma descrição ilustrativa em pseudo código, onde S é o conjunto solução da heurística e V é o conjunto de variáveis. Como visto no capítulo 3, o número de variáveis do modelo é $O(|P|^4)$. Além disso, devido à restrição de cardinalidade da seção 4.4.2 vemos que $|S| = O(|P|)$ e, como o teste de verificação de interseção estendida pode ser realizado em tempo constante, é possível observar que o tempo de execução da heurística descrita abaixo é $O(|P|^5)$.

1. $S \leftarrow \emptyset$
2. *Ordena* V
3. *Para cada* $v \in V$ *faça*
4. *Se para todo* $s \in S$, $R_s \wedge R_v = \emptyset$ *então*
5. $S \leftarrow S \bigcup \{v\}$
6. *Se (Linhas Cobertas* $= (|P|+1)^2$ *) então pare*

Três critérios de ordenação foram usados. No primeiro as variáveis são ordenadas de acordo com seus custos lagrangeanos. No segundo, são ordenadas de acordo com o valor da divisão do custo lagrangeano pela área do retângulo associado e, por fim, são ordenadas decrescentemente de acordo com custos (c'_j) calculados como no algoritmo do volume [20]. O custo é calculado a cada iteração do método do subgradiente da seguinte forma: $c'_j[i] = c'_j[i-1](1-\sigma)+\sigma y_j[i]$, onde i é a iteração corrente e σ é um fator cujo valor é 0,5. Os testes realizados levaram à escolha do primeiro critério, já que este obteve melhores resultados.

Essa heurística é chamada nas cem primeiras iterações, após o que ela só é chamada nas iterações pares.

5.5 Geração de Desigualdades Válidas

Nesta seção serão apresentados os algoritmos de geração de desigualdades que são violadas pela solução corrente do problema lagrangeano. Também serão explicados os critérios de gerenciamento de inclusão e exclusão destas restrições.

Na seção 4.4.1 está descrita a construção do grafo de interseção estendida. Neste grafo existe um vértice para cada retângulo viável da instância do problema. Mesmo para problemas pequenos, o número de vértices e de arestas deste grafo, torna proibitivo o seu uso para obtenção de desigualdades de cliques maximais e de ciclos ímpares violadas.

Para resolver este problema, constrói-se um subgrafo $G_i = (V_i, E_i)$ de G a cada iteração do método do subgradiente. Seja $I = (R, P)$ a instância do problema e $S_i = \{y_{i1}, \ldots, y_{i(|P|+1)}\}$ a solução do problema relaxado na iteração i. A cada variável $y_{ij} \in S_i$ corresponde um retângulo viável R_{ij} e um vértice $v_{ij} \in V_i$. O conjunto de arestas é definido como se segue: $E_i = \{(v_{ij}, v_{il}) \in V_i \mid R_{ij} \wedge R_{il} \neq \emptyset\}$. É sobre este grafo que são separadas as desigualdades válidas. Como o número de variáveis do modelo é $O(|P|^4)$, é possível observar que, no pior caso, a cardinalidade de G_i é $(O(|P|), O(|P|^2))$, enquanto que o grafo de interseção estendida completo G tem cardinalidade $(O(|P|^4), O(|P|^8))$.

O procedimento de separação de cortes (ver apêndice A) é chamado em toda iteração do método do subgradiente, e adiciona desigualdades válidas à formulação original. No entanto, o número potencialmente grande de desigualdades adicionadas pode causar problemas no cálculo do subgradiente, se aplicarmos diretamente as fórmulas da seção 5.3, o que traz a necessidade de alguns cuidados extras. Ilustrando melhor o problema, para cada iteração do método do subgradiente, podemos dividir as desigualdades geradas em três grupos. O primeiro é formado pelas desigualdades que são violadas por uma solução corrente do problema relaxado. O segundo contém desigualdades cujos multiplicadores são maiores que zero. Note que podemos ter interseção não vazia entre esses dois conjuntos. Por fim, o terceiro grupo é composto pelas desigualdades restantes. Outro ponto a se notar é que os multiplicadores que podem contribuir no cálculo do custo lagrangeano $\overline{c_j}$ são aqueles associados às desigualdades dos grupos um e dois. Por esse motivo, esses multiplicadores são chamados de *multiplicadores ativos* e, em oposição, denota-se os multiplicadores associados ao terceiro grupo de *multiplicadores inativos*. Nota-se ainda que o subgradiente associado a desigualdades não violadas é negativo, o que faz com que seu multiplicador correspondente aumente seu valor após a aplicação da fórmula de atualização. No caso específico das desigualdades do grupo três o multiplicador permanecerá inalterado em zero, já que, neste caso, este deve ser menor ou igual a zero.

Se por um lado multiplicadores inativos não contribuem para o cálculo do custo lagrangeano por outro, seus subgradientes associados influenciam bastante no tamanho de passo, tendendo a torná-los bastante pequenos, e isso faz com que todos os multiplicadores permaneçam praticamente inalterados na iteração subseqüente. Portanto, é necessário evitar um número muito grande de desigualdades inativas. Dessa maneira, aplica-se o método do subgradiente descartando-se as inequações do grupo três.

O resultado disto é um esquema dinâmico em que as desigualdades podem estar ativas ou inativas dependendo da iteração corrente do subgradiente. A introdução deste esquema

foi primeiramente proposta em [15], e em [17] este tipo de algoritmo foi chamado de *Relax and Cut.*

No algoritmo implementado, por causa de limitações de tempo e de memória da máquina, foi decidido não se manter um gerenciamento de conjuntos de desigualdades ativas e inativas. Assim, sempre que uma desigualdade se tornava inativa, esta era imediatamente eliminada da memória. No entanto, excetuando-se o tempo gasto nesse gerenciamento, não há perdas no esquema dinâmico, uma vez que desigualdades retiradas em uma iteração podem ser novamente geradas em iterações posteriores.

5.6 Fixação de Variáveis

No capítulo 4 são descritas algumas maneiras de diminuir o problema fixando, a priori, variáveis que garantidamente não fazem parte de uma solução ótima. Nesta seção será explicado como fixar variáveis durante a execução do algoritmo.

O valor da solução da relaxação lagrangeana do problema (Z_R) fornece um limite inferior para o valor da solução ótima do problema de PLI original (Z_{PI}), enquanto o valor da heurística primal (Z_{HP}) é um limite superior para este valor. Portanto, a seguinte relação vale:

$$Z_R \leq Z_{PI} \leq Z_{HP}. \tag{5.7}$$

Agora criamos um subproblema de minimização $Z_k = min\{\sum_{i=1}^{n} c_j y_j | y \in \Re_0 \bigcap \Re_1 \bigcap \Re_k\}$, onde $\Re_k$ é o hiperplano definido por $y_k = 1$. A resolução deste problema fornece um limite inferior para todas as soluções do problema tais que $y_k = 1$. Deste modo, tem-se a seguinte relação:

$$Z_{R_k} > Z_{PI} \Rightarrow y_k = 0. \tag{5.8}$$

Contudo, não se conhece o valor de Z_{PI}. Como é sábido que Z_{HP} é limite superior para este valor, se $Z_k > Z_{HP}$, então fixa-se y_k em zero.

Acima, vimos como fixar a variável y_k em zero, porém existe uma maneira de apertar ainda mais esse teste e, com isso, fixar mais variáveis. Seja Z'_{HP} o primeiro valor inteiro menor que Z_{HP} tal que possa existir uma solução viável para o problema com esse valor. Em geral, quando os valores dos custos da função objetivo são inteiros, tem-se que $Z'_{HP} = Z_{HP} - 1$. Mas neste caso específico, por causa do corolário 1, $Z'_{HP} = Z_{HP} - 2$. A partir disto, enunciamos o seguinte teorema.

Teorema 11 *Se* $Z_k > Z'_{HP}$ *então* $y_k = 0$.

Prova: A prova está dividida em dois casos.

- $Z'_{HP} \geq Z_{PI}$. *Neste caso, por hipótese,* $Z_{R_k} > Z'_{HP} \geq Z_{PI}$, *logo* $Z_{R_k} > Z_{PI}$, *o que satisfaz a condição 5.8.*

- $Z'_{HP} < Z_{PI}$. *Neste caso* $Z_{HP} = Z_{PI}$ *e já achamos uma solução ótima* S^*. *Então, se* $y_k \notin S^*$ *não há problemas em eliminá-la. Caso contrário,* $y_k \in S^*$ *e eliminá-la poderá aumentar o valor da solução do problema reduzido ou até mesmo torná-lo infactível. Em ambos os casos o valor da relaxação lagrangeana deve aumentar eventualmente passando o valor de* Z_{PI} *confirmando, assim, que* Z_{HP} *é uma solução ótima.* □

Na verdade, se o teste fixar uma variável que está na solução viável corrente do problema, então esta é uma solução ótima para este problema. Neste caso, o algoritmo pode parar tendo provado a otimalidade.

Duas maneiras foram criadas para obtenção de limites inferiores da solução do subproblema de minimização obtido pela fixação de uma variável.

1. Seja S_i a solução ótima do problema relaxado na iteração corrente. Seja y_j a variável de maior custo lagrangeano ($\overline{c_j}$) pertencente a S_i. Note que $Z_k \geq Z_R - \overline{c_j} + \overline{c_k}$. Se lembrarmos que os custos lagrangeanos já estão previamente calculados, para uma dada variável y_k, este procedimento tem tempo de execução constante. Na prática pode-se determinar um valor $Z_{HP} - (Z_R - \overline{c_j})$, tal que toda variável que possua custo lagrangeano maior que este valor, pode ser fixada em zero.

2. Eliminam-se todas as variáveis cujos retângulos tem interseção estendida com o retângulo associado a y_k. Desta maneira, pode-se dizer que $Z_k \geq \sum_{j \in J} \overline{c_j} + \overline{c_k}$, onde J é o conjunto das $|P|$ variáveis de menor custo lagrangeano. Para a execução deste algoritmo é necessário, no pior caso, testar interseção estendida do retângulo R_k com todos os outros do modelo. Isso pode ser conseguido em $O(|P|^4)$.

É fácil ver que o valor obtido pelo teste do item 2 obtém valores sempre maiores ou iguais aos obtidos no item 1.

Experimentalmente observamos que é mais interessante tentar fixar um número grande de variáveis em poucas chamadas, do que chamar várias vezes esse procedimento fixando poucas variáveis. Para refletir este princípio, tentamos fixar variáveis somente quando há uma melhora global no limite inferior, e desde que a diferença entre os limites superior e inferior seja menor que a metade do perímetro do retângulo envoltório, o que é aproximadamente o valor da variável de maior custo. Apesar de ser mais custoso, utilizamos o segundo critério citado acima já que este é bem mais efetivo e com isso o tempo ganho na relaxação,

durante o cálculo do subgradiente e na heurística, compensam o tempo extra gasto nessa etapa. Além disso, adotamos a seguinte estratégia dentro do procedimento de fixação de variáveis: inicialmente tentamos fixar alguma variável no conjunto das 5% de maior custo lagrangeno, se obtivermos sucesso, continuamos com os próximos 5% até que se complete um máximo de 30% do total de variáveis testadas.

5.7 Tópicos de Implementação

A concepção inicial deste projeto era ter um código bastante flexível que pudesse absorver as possibilidades de testes e de mudanças no curso do trabalho. Além disso, ainda existia a possibilidade de resolução de outros problemas mais gerais de partição de conjuntos com o *Relax & Cut*. Essa visão genérica sempre foi um guia, no entanto não era o objeto principal do nosso estudo. Dessa maneira esta seção não tem a pretensão de levar o assunto à exaustão, mas sim de deixar registradas as experiências adquiridas nesta tarefa.

Primeiro resolvemos utilizar uma metodologia baseada em orientação a objetos, pois nos parece ser a que traz maior flexibilidade e facilidade de implementações opcionais, caso o modelo espelhe suficientemente bem a realidade e a implementação seja fiel a este modelo. A figura 5.2 mostra o diagrama de classes na notação UML[1] de [26]. Esse diagrama apresenta uma versão reduzida, uma vez que não estão representados os métodos e os atributos.

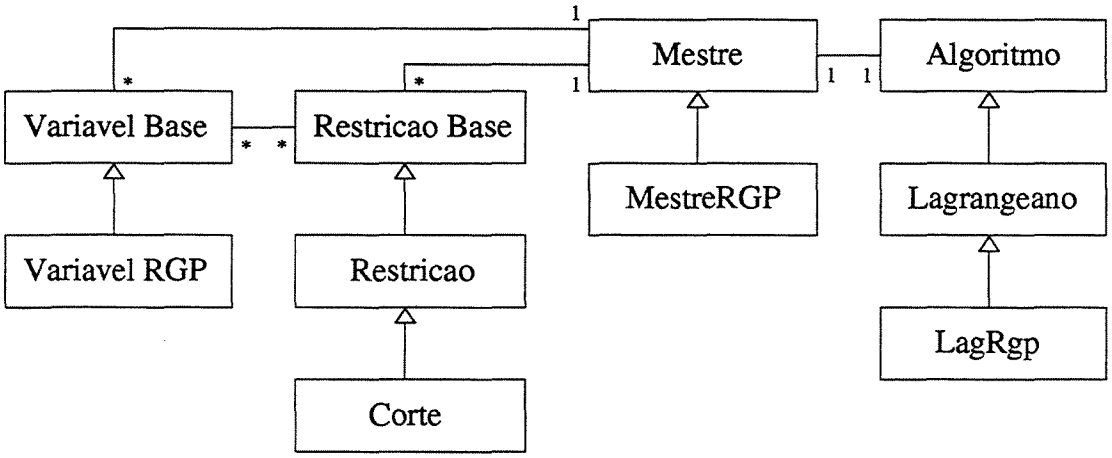


Figura 5.2: Modelo de classes de problemas de otimização combinatória usando algoritmos lagrangeanos.

A idéia básica por trás desse modelo é separar as estruturas de dados e os gerenciamentos

[1]Unified Modelling Language

mais básicos da resolução do problema. Isso é feito com as classes *Mestre* e *Algoritmo*, como será esclarecido mais adiante. Outro ponto é a definição bem precisa das classes que espelham mais a realidade de modo que estas sejam auto-contidas e tenham como fornecer ao máximo as informações que se necessitam delas. Na verdade, isto é mais uma maneira de separar as tarefas de cada componente, o que é a intenção das metodologias orientadas a objeto.

A classe *Mestre* é responsável pelo gerenciamento do problema como um todo. Ela vai ter o conjunto de todas as variáveis, bem como os diversos conjuntos de restrições que podem estar presentes no problema, como, por exemplo, o conjunto de restrições originais dualizadas e o conjuntos de cortes válidos. Uma classe específica derivada de *Mestre*, que no caso é a classe *MestreRGP*, é responsável pela geração ou carga na memória do problema em questão.

A classe *Algoritmo* representa o método que será utilizado para a resolução do problema, e como pode ser visto no modelo, este poderia até mesmo ser um resolvedor de programação linear ou uma heurística Tabu. Apesar disto, implementou-se apenas o algoritmo lagrangeano. Este é modelado nas classes derivadas *Lagrangeano* e *LagRGP*. A primeira é responsável pelas tarefas mais genéricas, como o cálculo do subgradiente e a separação de cortes, enquanto que a segunda responde pela relaxação lagrangeana e a fixação de variáveis, específicas do **RG-P**.

As classes *RestricaoBase* e *VariavelBase* representam, respectivamente, todas as restrições do problema, independente de serem cortes e de estarem ou não dualizadas, e todas as variáveis. No modelo podemos ver que estas têm um relacionamento de cardinalidade muitos para muitos. Na prática esse relacionamento é implementado da maneira que se segue. Para todas as restrições originais do problema temos a representação de colunas, ou seja, a variável tem um conjunto de restrições às quais ela pertence. Já os cortes possuem um conjunto de variáveis. Desta forma, não há maneira direta para uma dada variável saber a quais cortes pertence e, da mesma forma uma dada restrição original do problema não tem informações sobre quais variáveis ela contem. Esta implementação não fere o modelo proposto, e foi usada porque obteve melhor desempenho na inserção e remoção de cortes e no cálculo do subgradiente.

Utilizamos ainda a STL [28], que atualmente é uma biblioteca padrão nas distribuições do g++[2] para Linux. Essa biblioteca é composta de um conjunto de *templates*, ou tipos paramétricos de dados. Na prática são classes que fazem suas operações sobre um tipo que será passado como parâmetro pelo usuário. Um bom exemplo seria o *template* de um vetor, que teria implementado todas as funções básicas de inserção e remoção, bem como algoritmos de ordenação, e dessa maneira, uma simples declaração de vetor de variáveis ou de restrições já proporcionaria essas facilidades para as referidas classes. Isso trouxe,

[2]g++ é o compilador C++ da gnu.

portanto, um ganho no tempo de desenvolvimento, já que muitos algoritmos complexos já estão implementados nesta biblioteca. Além disso ainda houve bastante ganho no que diz respeito à padronização.

O exposto até o momento leva à conclusão de que a implementação de um método para a resolução de outro problema de otimização combinatória passará no máximo pela construção de novas classes que substituam *VariavelBase, LagRGP, MestreRGP* e, dependendo do caso, *Restricao e Corte.* Além disso, o teste de vários algoritmos de fixação de variáveis ou de obtenção de soluções primais viáveis fica bastante facilitado, uma vez que estes são bastante independentes. Outro aspecto é que a utilização de um novo corte pode ser feita apenas com a construção de uma nova classe específica, sem que para isso precise ser feita qualquer alteração em outros pontos do código.

Capítulo 6

Experimentos Computacionais

Neste capítulo serão apresentados os resultados computacionais obtidos neste trabalho. A heurística lagrangeana descrita nos capítulos anteriores foi implementada em C++. Utilizamos ainda a *STL* [28] para auxiliar na implementação das estruturas de dados e dos algoritmos mais comuns. Também usamos o CPLEX (versão 6.5) [27] para a resolução de problemas de programação linear inteira. Os testes computacionais foram realizados em um Pentium III de 450 MHz e 384 MB de memória RAM[1].

Nas seções subsequentes serão descritos e apresentados mais detalhadamente os testes realizados. Inicialmente descrevemos as instâncias utilizadas. Em seguida mostramos os resultados obtidos com a eliminação de variáveis utilizando propriedades geométricas. Mostramos ainda os resultados obtidos pelo CPLEX quando este resolve diretamente o problema e, finalmente, apresentamos os resultados obtidos pela heurística lagrangeana e algumas variações desta.

6.1 Instâncias

Os testes foram realizados com instâncias de pontos não corretilineares que podem ser divididos em três conjuntos principais. O primeiro conjunto é o mesmo utilizado em [9], onde o número de pontos terminais tem cardinalidade entre 15 e 39 e o retângulo envoltório tem dimensões de 20 por 20 ou 50 por 50. O segundo conjunto, que chamamos de instâncias de médio porte, tem entre 40 e 60 pontos e retângulo envoltório com dimensão 100 por 100. O terceiro conjunto é formado por instâncias contendo de 80 a 90 pontos terminais e retângulo envoltório com dimensão 100 por 100. Classificamos este último conjunto como sendo instâncias de grande porte.

Para obter as coordenadas dos pontos de P, usamos o mesmo gerador de números pseudo

[1] Random Acess Memory

UNICAMP
BIBLIOTECA CENTRAL
SEÇÃO CIRCULANTE

aleatórios com distribuição uniforme, utilizado em [9]. Este programa sempre gera pontos com coordenadas inteiras.

6.2 Resultados das Reduções Geométricas

Nesta seção mostramos as principais características das instâncias, como por exemplo, o número de linhas e de colunas do modelo de PLI descrito no capítulo 3, além dos resultados obtidos com a redução de variáveis descrita no capítulo 4.

Os resultados são apresentados nas tabelas 6.1 e 6.2 cujas colunas podem ser descritas como se segue: $|P|$ é a cardinalidade do conjunto de pontos; *Linhas* é o número de linhas da instância; *Cols* é o número de colunas; *Cols Red1* é o número de colunas após a redução da seção 4.1; *Cols Red2* é o número de colunas restantes após a redução da seção 4.2; *% Red* é a porcentagem de colunas que restam no problema; *Não Nulos* é o número de elementos não nulos da matriz de restrições e *Dens* é a densidade da matriz do problema reduzido.

Podemos observar nas tabelas, que a redução obtida pela aplicação das propriedades geométricas foi bastante grande, conseguindo reduzir o número de variáveis a aproximadamente um terço do original. Além disso, observa-se também, uma leve tendência dessas reduções serem proporcionalmente ainda mais efetivas, quando aumentamos o número de pontos. Uma inspeção mais cuidadosa das instâncias testadas mostra que, apesar de teoricamente o número de variáveis ser $O(|P|^4)$, após as reduções esse número tem valor bastante próximo a $|P|^3$, o que pode ser observado no gráfico da figura 6.1.

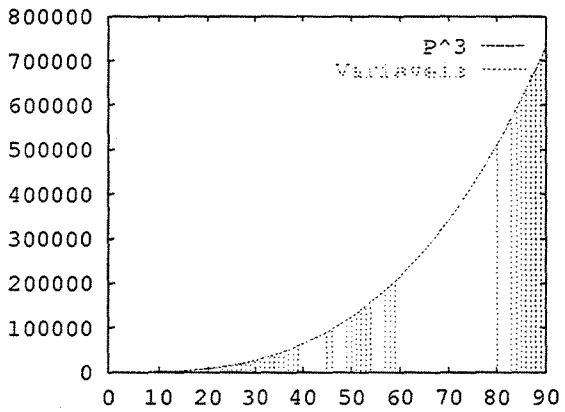


Figura 6.1: Variáveis em função do número de pontos da instância, comparado com $|P|^3$

Outro dado que merece atenção, é que a densidade da matriz de restrições do problema vai caindo conforme aumenta o número de pontos em P.

$\|P\|$	Linhas	Cols	Cols Red1	Cols Red2	% Red	Não Nulos	Dens
15	256	7262	5767	2810	38.69%	36407	5.06%
16	289	9292	7438	3356	36.12%	48336	4.98%
17	324	11131	9075	4075	36.61%	62648	4.74%
18	361	13319	10852	4866	36.53%	77150	4.39%
19	400	15994	13202	5821	36.39%	98917	4.25%
20	441	18731	15638	6707	35.81%	119963	4.06%
21	484	22214	18721	7941	35.75%	151921	3.95%
22	529	26286	22278	9053	34.44%	185805	3.88%
23	576	28434	24116	10440	36.72%	204933	3.41%
24	625	32855	28052	12046	36.66%	248394	3.30%
25	676	37952	32613	13600	35.83%	298578	3.25%
26	729	43508	37608	15295	35.15%	352241	3.16%
27	784	47767	41531	17276	36.17%	399972	2.95%
28	841	54646	47677	19492	35.67%	476564	2.91%
29	900	62133	54544	21367	34.39%	553183	2.88%
30	961	69612	61472	24371	35.01%	662948	2.83%
31	1024	76269	67442	26866	35.23%	731061	2.66%
32	1089	85703	76156	28901	33.72%	830162	2.64%
33	1156	94713	84253	32416	34.23%	956873	2.55%
34	1225	104149	93425	35260	33.86%	1104910	2.56%
35	1296	116590	104952	38505	33.03%	1271941	2.55%
36	1369	123787	111309	42282	34.16%	1343419	2.32%
37	1444	149378	135534	47007	31.47%	1778684	2.62%
38	1521	143874	130125	50064	34.80%	1648905	2.17%
39	1600	157432	142751	54011	34.31%	1835457	2.12%
45	2116	258502	237508	84890	32.84%	3472641	1.93%
46	2209	265395	243804	92472	34.84%	3638703	1.78%
49	2500	351298	325099	111557	31.76%	5218077	1.87%
50	2601	369469	342116	118187	31.99%	5470033	1.78%
53	2916	446723	415282	142444	31.89%	6994448	1.68%
54	3025	508815	476591	147344	28.96%	8225872	1.85%
57	3364	551419	515173	179106	32.48%	9261540	1.54%
58	3481	609384	572002	184283	30.24%	10338239	1.61%
60	3721	652141	610827	208216	31.93%	11309285	1.46%

Tabela 6.1: Redução de colunas. Instâncias de pequeno e médio porte.

$\|P\|$	Linhas	Cols	Cols Red1	Cols Red2	% Red	Não Nulos	Dens
80	6561	1729832	1648992	514078	29.72%	39679724	1.18%
83	7056	1851493	1765696	565031	30.52%	42384352	1.06%
84	7225	1932570	1844715	593793	30.73%	45271553	1.06%
85	7396	2047803	1955262	619038	30.23%	48487932	1.06%
86	7569	2045611	1952451	637382	31.16%	48377078	1.00%
87	7744	2160537	2064750	667367	30.89%	52735326	1.02%
88	7921	2225525	2128110	682492	30.67%	54023955	1.00%
89	8100	2208191	2110458	698859	31.65%	52676355	0.93%
90	8281	2455502	2351651	721210	29.37%	59628674	1.00%

Tabela 6.2: Redução de colunas. Instâncias de grande porte.

6.3 Resultados Obtidos com o CPLEX

Nesta seção descrevemos os resultados obtidos ao utilizarmos o CPLEX [27] para resolver as instâncias de teste do **RG-NLP**. Os problemas de PLI resolvidos aqui já são aqueles após as reduções geométricas de variáveis mostradas na seção anterior. O CPLEX é reconhecido como um dos melhores e mais rápidos resolvedores profissionais de PLI do mercado.

Na tabela 6.3 as colunas têm os seguintes significados: $|P|$ é a cardinalidade do conjuntos de pontos da instância; *Tempo Relax* é o tempo da resolução da relaxação linear no nó raiz da árvore de Branch & Bound; *Ótimo Relax* é o valor da solução ótima da relaxação linear; *Tempo CPLEX* é o tempo gasto para encontrar a solução ótima inteira; *Ótimo Inteiro* é o valor da melhor solução inteira; *Iter* é o número de iterações do método simplex; *Nós do B&B* é o número de nós gerados durante a execução do Branch & Bound e *GAP* é a diferença percentual entre o ótimo da relaxação linear e o ótimo inteiro. Para facilitar a visualização, o resultado da coluna GAP será omitido sempre que este valer zero.

Observando-se o *GAP* de integralidade, nota-se que na grande maioria dos casos, o valor da solução ótima da relaxação linear é o mesmo da solução inteira. E quando isto não ocorre, o simples arredondamento para o próximo número par muitas vezes faz atingir este valor. [2] Esses resultados mostram características bem particulares deste problema e reafirmam os valores já reportados em [9].

Ainda na tabela 6.3, podemos observar que somente são apresentados os resultados para instâncias com até 50 pontos. Isso ocorre porque, apesar da resolução do problema ter sido relativamente rápida (aproximadamente 20 min), não havia memória suficiente para a resolução de instâncias maiores. Nas próximas seções veremos as tentativas que fizemos para contornar essas dificuldades.

[2] Como visto no teorema 3, a solução ótima inteira é sempre par.

$\|P\|$	Tempo Relax	Ótimo Relax	Tempo CPLEX	Ótimo Inteiro	Iter	Nós do B&B	GAP
15	0.17	380	0.66	380	226	—	—
16	0.29	386	0.96	386	291	—	—
17	0.45	395.33	4.14	398	408	2	0.67%
18	0.88	408	2.21	408	424	—	—
19	0.85	414	2.89	414	415	—	—
20	0.69	1022	3.32	1022	383	—	—
21	1.08	1046	4.96	1046	397	—	—
22	2.11	1062	16.47	1062	569	1	—
23	1.34	1052	7.06	1052	464	—	—
24	3.58	1134	11.18	1134	724	—	—
25	4.30	1112	14.75	1112	801	—	—
26	3.87	1142	17.05	1142	716	—	—
27	8.68	1140	24.37	1140	1010	—	—
28	6.60	1156	28.50	1156	797	—	—
29	11.02	1174	67.98	1174	1002	—	—
30	12.51	1174	50.30	1174	1076	—	—
31	21.86	1220	64.06	1220	1463	—	—
32	14.44	1220	64.65	1220	1199	—	—
33	18.47	1246	80.03	1246	1327	—	—
34	16.81	1236	94.78	1236	1326	—	—
35	52.58	1269	217.80	1270	2366	2	0.08%
36	39.37	1279	176.73	1280	1821	2	0.08%
37	95.06	1278	286.49	1278	2791	—	—
38	104.50	1314	329.60	1314	3022	—	—
39	89.42	1324	260.22	1324	2715	—	—
45	225.60	2749	711.09	2752	4302	2	0.11%
46	195.48	2794	643.29	2794	3570	—	—
49	171.30	2802	871.57	2802	3468	—	—
50	468.76	2833	1258.27	2834	5760	9	0.04%

Tabela 6.3: Resultados obtidos com o CPLEX

6.4 Resultados Obtidos com a Heurística Lagrangeana

Serão apresentados nesta seção os resultados obtidos com algumas variações da heurística lagrangeana. Algumas considerações iniciais são necessárias para a boa compreensão dos resultados.

Como podemos observar na seção anterior, o algoritmo usado pelo CPLEX é extremamente eficiente para instâncias pequenas esbarrando, contudo, no limite de memória disponível quando procuramos tratar instâncias maiores. Logo, nosso objetivo principal é possibilitar a resolução de instâncias maiores. Logicamente, a variável tempo não pode ser desprezada e, se possível deve também ser melhorada.

Nossa estratégia, então, será desenvolver um método que aproveite a parte boa de cada um dos dois algoritmos. A capacidade de obter bons limites e fixar variáveis da heurística lagrangeana, aliada à capacidade de resolução extremamente precisa e rápida, que o CPLEX possui para instâncias pequenas. Chamamos esse algoritmo de *método híbrido*.

6.4.1 Heurística Lagrangeana sem Cortes

A primeira tentativa de uso da Heurística Lagrangeana foi executá-la sem a geração de cortes e com os parâmetros descritos no capítulo 5, ou seja, com 2000 iterações e $\gamma = 80$.

As tabelas 6.4 e 6.5 apresentam os resultados da execução da heurística. A coluna $|P|$ é a cardinalidade do conjunto de pontos; *LS* apresenta o valor da melhor solução inteira obtida (Limite superior); *LI* apresenta o melhor valor da Relaxação Lagrangeana (Limite inferior); *GAP* é a diferença percentual entre a *LS* e *LI*; *Tempo* é o tempo gasto em segundos para a execução do algoritmo; *Iter* é o número de iterações do método subgradiente; *Var Rest* é o número de variáveis restantes após a execução do algoritmo; *% Var Fix* apresenta a porcentagem de variáveis fixadas desde o início da execução (número de variáveis fixadas dividido pela número de colunas após as reduções geométricas) e *% Var Fix2* é a porcentagem de variáveis fixadas em relação à formulação original (variáveis fixadas dividido pelo número de colunas antes das reduções). Para facilitar a compreensão, os resultados das colunas *LI* e *GAP* são omitidos sempre que a Heurística Lagrangeana provar otimalidade para o problema.

Comparando a tabela 6.4 com a tabela 6.3 podemos ver resultados bastante interessantes. Depois da instância com 29 pontos, todas as que o CPLEX resolveu foram mais lentas que a Heurística (vide figura 6.3), muito embora a Heurística não tenha conseguido provar otimalidade em todos os casos. Mas isso não parece ser um grande problema, já que após a sua execução as instâncias ficaram com menos de 5000 variáveis e, espera-se que o CPLEX (segundo passo do método híbrido) os resolva em menos de 10 segundos. De fato, esses resultados serão comprovados mais adiante.

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
15	380	—	—	1.12	399	119	95.77%	98.36%
16	386	—	—	1.67	511	125	96.26%	98.65%
17	398	395.2869	0.68%	3.94	2000	198	95.14%	98.22%
18	408	—	—	4.15	795	110	97.74%	99.17%
19	414	—	—	4.16	543	363	93.76%	97.73%
20	1022	—	—	5.80	690	258	96.15%	98.62%
21	1046	—	—	7.39	742	495	93.77%	97.77%
22	1062	—	—	8.24	1157	60	99.34%	99.77%
23	1052	—	—	5.53	436	182	98.26%	99.36%
24	1134	—	—	19.04	1134	123	98.98%	99.63%
25	1112	—	—	15.61	719	450	96.69%	98.81%
26	1142	—	—	19.08	827	163	98.93%	99.63%
27	1140	—	—	23.83	770	164	99.05%	99.66%
28	1156	—	—	31.29	833	151	99.23%	99.72%
29	1174	—	—	41.01	1103	113	99.47%	99.82%
30	1174	—	—	42.40	1057	231	99.05%	99.67%
31	1220	—	—	48.91	1278	109	99.59%	99.86%
32	1220	—	—	54.40	1012	167	99.42%	99.80%
33	1246	—	—	71.48	1041	503	98.45%	99.47%
34	1236	—	—	59.31	1236	1761	95.01%	98.31%
35	1270	1266.7919	0.25%	89.07	2000	698	98.19%	99.40%
36	1280	—	—	84.71	1633	140	99.67%	99.89%
37	1282	1275.5851	0.50%	156.65	2000	4438	90.56%	97.03%
38	1314	—	—	88.49	1305	185	99.63%	99.87%
39	1326	1321.2452	0.36%	141.12	2000	2015	96.27%	98.73%
45	2752	2746.9338	0.18%	215.76	2000	1338	98.42%	99.48%
46	2794	—	—	209.90	1226	255	99.72%	99.90%
49	2810	2801.2864	0.31%	514.40	2000	3609	96.76%	98.97%
50	2834	2828.6609	0.19%	450.98	2000	2046	98.27%	99.45%
53	2960	2936.4180	0.80%	902.39	2000	39881	72.00%	91.07%
54	2896	2859.5322	1.26%	1119.57	2000	58639	60.20%	88.48%
57	3076	3051.8574	0.78%	1130.67	2000	50433	71.84%	90.85%
58	3040	3019.2156	0.68%	1352.76	2000	44354	75.93%	92.72%
60	3092	3082.9722	0.29%	1022.41	2000	11031	94.70%	98.31%

Tabela 6.4: Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes.

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
80	3462	3447.0022	0.43%	2935.88	2000	63246	87.70%	96.34%
83	3548	3524.7271	0.66%	3737.07	2000	147292	73.93%	92.04%
84	3584	3544.6189	1.10%	4068.00	2000	265365	55.31%	86.27%
85	3610	3548.0371	1.72%	4234.63	2000	400159	35.36%	80.46%
86	3650	3600.7646	1.35%	4623.12	2000	351888	44.79%	81.79%
87	3678	3637.7058	1.10%	4198.42	2000	320253	52.01%	85.18%
88	3652	3631.9907	0.55%	4686.37	2000	151949	77.74%	93.17%
89	3736	3699.9316	0.97%	4551.24	2000	286710	58.97%	87.02%
90	3678	3645.5200	0.88%	4439.60	2000	270570	62.48%	88.98%

Tabela 6.5: Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes. Instâncias de grande porte.

Olhando os resultados da tabela 6.5, vemos que, com a quantidade de memória disponível, a Heurística Lagrangeana consegue tratar instâncias com até 90 pontos, o que significa aproximadamente 7 vezes mais variáveis que a maior instância tratada pelo CPLEX (vide tabelas 6.1 e 6.2). Apesar disto, a resolução até a otimalidade depende bastante de quantas variáveis são fixadas durante a execução, o que aparentemente fica cada vez mais difícil a medida que cresce o número de pontos. Isto pode ser explicado, em parte, pela observação de que as variáveis começam a ser fixadas, quando a diferença entre o limite superior e o inferior é aproximadamente metade do perímetro do retângulo envoltório (aproximadamente o valor da variável de maior custo) e, já que o custo da solução ótima cresce conforme o número de pontos, a diferença percentual entre os limites superior e inferior deve ser cada vez menor para que o processo de fixação ocorra.

Na tabela 6.6 são apresentados os resultados da execução do CPLEX com o problema que sobra após a execução da Heurística Lagrangeana. As colunas seguem o mesmo padrão de nomes da tabela 6.3. Nesses testes foi possível resolver todas as instâncias até 60 pontos e até mesmo uma instância de grande porte (80 pontos). Instâncias maiores que 80 pontos não puderam ser resolvidas devido a limites de memória. Outro ponto positivo que vale ser ressaltado é que, mesmo quando a resolução do problema não foi possível, a qualidade da solução obtida pela relaxação lagrangeana é excelente. Este fato pode ser ilustrado notando-se que a instância com 85 pontos apresenta o maior *GAP*, que é apenas de 1.72%. Além disso, todos os resultados obtidos com a heurística lagrangeana para essas instâncias foram executados em menos de 80 minutos e, para instâncias entre 29 e 50 pontos, o algoritmo híbrido foi mais rápido que o CPLEX quando executado sozinho.

$\|P\|$	Tempo Relax	Ótimo Relax	Tempo CPLEX	Ótimo Inteiro	Iter	Nós do B&B	GAP
35	0.25	1269	1.72	1270	434	2	0.08%
37	5.29	1278	8.20	1278	1283	—	—
39	1.65	1324	2.20	1324	870	—	—
45	0.71	2749	2.86	2752	635	2	0.11%
49	2.69	2082	3.81	2082	1065	—	—
50	1.49	2833	7.84	2834	835	—	0.04%
53	417.70	2947	922.69	2948	8946	2	0.03%
54	243.27	2868.66	802.84	2870	5689	2	0.05%
57	454.55	3066	664.94	3066	9454	—	—
58	505.99	3027.33	1837.14	3036	21304	65	0.29%
60	65.76	3091	142.16	3092	4603	2	0.03%
80	1373.35	3456	1718.16	3456	19911	—	—

Tabela 6.6: Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana sem cortes

6.4.2 Heurística Lagrangeana com Cortes

A Heurística da forma como implementada na seção anterior, teve grande sucesso na resolução de instâncias de médio porte e obteve excelentes limites na execução de instâncias grandes. Mas, como explicado anteriormente, para as instâncias maiores, o número de variáveis restante após a sua execução ainda era muito grande para permitir que o CPLEX o resolvesse até a otimalidade.

A idéia então é tentar aumentar o limite inferior para que fossem fixadas mais variáveis. Isto foi feito justamente com a inserção de planos de corte automaticamente dualizados (Relax & Cut), como descrito na seção 5.5. Nas tabelas 6.7 e 6.8 podemos encontrar os resultados obtidos por esse método, as colunas desta tabela seguem o mesmo padrão de nomes da tabela 6.4.

Para as instâncias pequenas e médias o algoritmo com cortes foi mais lento que o anterior. No entanto, este ainda executa mais rapidamente que o CPLEX para quase todas as instâncias a partir de 29 pontos, como pode ser visto na figura 6.3.

Para as instâncias maiores, observamos uma tendência da heurística com cortes obter melhores limites, fixar mais variáveis e executar em menos tempo. Como pode ser visto na comparação entre as 9 instâncias das tabelas 6.5 e 6.8, apenas um limite inferior obtido pela heurística com cortes é pior, em três casos esta é mais lenta que a heurística sem cortes e o número de variáveis fixadas só perde para a heurística sem cortes em dois casos. Apesar dos resultados obtidos com a inserção de cortes se mostrarem melhores, ao contrário do

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
15	380	—	—	1.99	555	189	93.27%	97.40%
16	386	—	—	2.27	442	200	94.04%	97.85%
17	398	—	—	5.25	1254	93	97.72%	99.16%
18	408	—	—	4.55	705	110	97.74%	99.17%
19	414	—	—	6.11	562	216	96.29%	98.65%
20	1022	—	—	8.02	804	222	96.69%	98.81%
21	1046	—	—	6.76	614	219	97.24%	99.01%
22	1062	—	—	11.08	749	84	99.07%	99.68%
23	1052	—	—	9.11	521	260	97.51%	99.09%
24	1134	—	—	23.53	1345	89	99.26%	99.73%
25	1112	—	—	28.07	1043	217	98.40%	99.43%
26	1142	—	—	28.09	997	267	98.25%	99.39%
27	1140	—	—	33.78	1084	180	98.96%	99.62%
28	1156	—	—	25.35	983	255	98.69%	99.53%
29	1174	—	—	60.19	1388	246	98.85%	99.17%
30	1174	—	—	30.38	972	218	99.11%	99.69%
31	1220	—	—	75.03	1277	219	99.18%	99.71%
32	1220	—	—	65.76	1041	679	97.65%	99.21%
33	1246	—	—	99.16	1432	657	97.97%	99.31%
34	1236	—	—	60.43	819	1229	96.51%	98.82%
35	1270	—	—	134.76	2000	487	98.74%	99.58%
36	1280	1277.4324	0.20%	102.39	2000	690	98.37%	99.44%
37	1282	1273.9564	0.63%	198.21	2000	7653	83.72%	94.88%
38	1314	—	—	187.34	1491	438	99.13%	99.70%
39	1326	1321.6467	0.33%	215.84	2000	486	99.10%	99.69%
45	2752	2748.7854	0.12%	291.77	2000	912	98.93%	99.65%
46	2794	—	—	272.01	1426	386	99.58%	99.85%
49	2814	2799.9705	0.50%	517.21	2000	14252	87.22%	95.94%
50	2834	2827.897	0.22%	481.24	2000	4411	96.27%	98.81%
53	2954	2936.8979	0.58%	1003.89	2000	26935	81.09%	93.97%
54	2872	2861.3911	0.37%	913.67	2000	11723	92.04%	97.70%
57	3096	3053.147	1.38%	1397.04	2000	90813	49.30%	83.53%
58	3050	3016.894	1.09%	1424.95	2000	79548	56.83%	86.95%
60	3092	3082.876	0.30%	1224.48	2000	13465	93.53%	97.94%

Tabela 6.7: Resultados obtidos com a Heurística Lagrangeana com o uso de cortes.

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
80	3456	3447.5120	0.25%	2511.12	2000	23525	95.42%	98.64%
83	3544	3526.1497	0.50%	3267.08	2000	112156	80.15%	93.94%
84	3578	3542.8313	0.98%	4484.13	2000	254435	57.15%	86.83%
85	3600	3550.5642	1.37%	4176.16	2000	347066	43.93%	83.05%
86	3644	3601.5862	1.16%	4478.02	2000	314660	50.63%	84.62%
87	3678	3638.4788	1.07%	4653.46	2000	319859	52.07%	85.20%
88	3654	3632.6797	0.58%	4545.54	2000	178815	73.80%	91.97%
89	3742	3700.6572	1.10%	4311.14	2000	332736	52.39%	84.53%
90	3678	3646.2471	0.86%	5119.90	2000	276103	61.72%	88.76%

Tabela 6.8: Resultados obtidos com a Heurística Lagrangeana com o uso de cortes. Problemas de grande porte.

$\|P\|$	Tempo Relax	Ótimo Relax	Tempo CPLEX	Ótimo Inteiro	Iter	Nós do B&B	GAP
36	0.89	1280	1.07	1280	492	—	—
37	35.25	1278	43.65	1278	2867	—	—
39	0.39	1324	0.58	1324	312	—	—
45	1.57	2752	1.86	2752	665	—	—
49	43.69	2802	58.75	2802	2938	—	—
50	16.75	2834	18.68	2834	2030	—	—
53	287.55	2948	351.47	2948	7165	—	—
54	106.00	2870	119.16	2870	4985	—	—
57	999.40	3066	1619.05	3066	8458	—	—
58	1221.95	3029.32	5106.07	3036	20859	31	0.22%
60	177.10	3092	293.46	3092	6751	—	—
80	355.50	3456	401.29	3456	8471	—	—

Tabela 6.9: Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana com cortes

que foi conseguido na seção anterior com a heurística sem cortes, em relação ao CPLEX, não pudemos encontrar uma instância a partir da qual a inserção de cortes se torna sempre vantajosa. Apesar disto, uma comparação das tabelas 6.6 e 6.9 mostra que os cortes adicionados ao problema sempre conseguiram fazer com que o GAP de integralidade diminuísse, na maioria das vezes caindo para zero. Isso resultou num menor número de nós na árvore de procura do *Branch & Bound* e comprovou a utilidade dos cortes gerados.

$\|P\|$	Cortes Inseridos	Cortes Removidos	Cortes Finais
17	7214	6878	336
35	18783	18147	636
45	24375	23286	1089
58	34161	32901	1260
83	40693	38271	2422
89	47140	45378	1762

Tabela 6.10: Cortes gerados durante a execução da heurística lagrangeana.

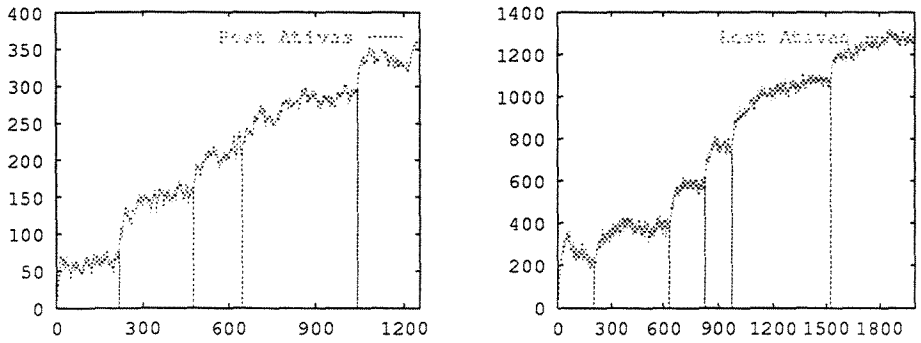


Figura 6.2: Número de desigualdades ativas ao longo das iterações. As linhas verticais representam o ponto onde o parâmetro α é reduzido. (a) Instância com 17 pontos. (b) Instância com 58 pontos.

A eficiência dos cortes que estão presentes após a execução da heurística lagrangeana foi discutida acima. A tabela 6.10 apresenta uma visão mais dinâmica dos cortes gerados durante a execução do algoritmo lagrangeano. Nesta tabela podemos ver o número de desigualdades que são inseridas, quantas são removidas e quantas restam ao final. Podemos notar que a grande maioria das desigualdades geradas se torna desnecessária em iterações posteriores e, por isso, são eliminadas.

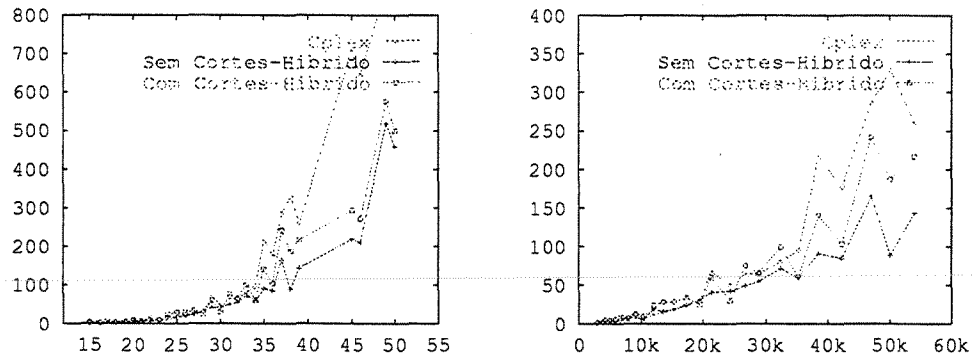


Figura 6.3: (a) Tempo em função do número de pontos da instância. (b) Tempo em função do número de variáveis.

Outra visão é a da figura 6.2, que mostra o número de restrições ativas ao longo das iterações. É possível observar que o gráfico tem grandes subidas seguidas por períodos de estabilidade. Examinando a execução do algoritmo observamos que essas subidas ocorrem quando o valor de α é reduzido à metade (Ver seção 5.3). Isto pode ser explicado pelo fato de que o limite inferior também cresce bastante nesse momento e, uma vez que os cortes são gerados a partir da solução do problema relaxado, eles acabam sendo mais eficientes e tendem a se manter até o final.

A expectativa de encontrar uma instância a partir da qual a heurística lagrangeana com cortes fosse melhor em todos os aspectos do que a heurística sem cortes não se realizou. No entanto, os resultados obtidos para as instâncias de grande porte mostraram que o uso dos cortes foram responsáveis por uma melhora de seis das oito instâncias de teste. Uma hipótese para a frustração desta expectativa pode vir do fato do *GAP* de integralidade ser bastante pequeno em todas as nossas instâncias. As próximas seções trarão experimentos que investigam essa hipótese.

6.4.3 Mudando os Parâmetros de Execução

A estratégia de inserção de cortes parece ajudar tanto na obtenção de soluções melhores, como na fixação de mais variáveis para as instâncias de grande porte. No entanto, essa melhora, ainda não foi suficiente para atingirmos o número de variáveis necessárias para a posterior execução do CPLEX.

A próxima tentativa é variar os parâmetros de execução do algoritmo para as instâncias maiores. O objetivo é dar mais tempo para ver se a relaxação lagrangeana encontra limites inferiores e superiores melhores. Desse modo também estaríamos estudando o comportamento das duas heurísticas quando o limite inferior se aproxima do ótimo da relaxação

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
80	3456	—	—	5209.28	5184	1003	99.80%	99.94%
83	3532	—	—	5490.18	4816	2891	99.49%	99.84%
84	3562	3556.7437	0.15%	6360.91	8000	9009	98.48%	99.52%
85	3572	3561.3384	0.30%	6601.48	8000	44534	92.81%	97.83%
86	3628	3612.4097	0.43%	8107.73	8000	97238	84.74%	95.35%
87	3666	3654.5254	0.31%	7668.26	8000	61280	90.82%	97.16%
88	3644	—	—	5612.90	8000	2082	99.69%	99.91%
89	3722	—	—	7318.70	8000	2755	99.61%	99.88%
90	3676	3657.8540	0.49%	10139.98	8000	129759	82.01%	94.72%

Tabela 6.11: Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes com γ=300 e 8000 iterações.

linear. A idéia, então, parece ser simplesmente aumentar o número de iterações do método do subgradiente. No entanto, observando os resultados da execução do algoritmo, vimos que o parâmetro α já tem valor muito pequeno quando o algoritmo se aproxima de 2000 iterações. Com isso, o tamanho de passo fica muito pequeno, o que faz a relaxação lagrangeana perder a força para achar melhores soluções e, portanto, o simples aumento no número de iterações não traz um aumento significativo do valor do limite inferior. Para contornar esse problema temos que alterar algum outro parâmetro. O parâmetro a ser alterado é o γ, que passa a ser 300 e, com isso, passemos a estabelecer um novo limite de 8000 iterações para a execução do algoritmo.

Os resultados da execução do algoritmo com os novos parâmetros podem ser vistos nas tabelas 6.11 e 6.14, que estão organizadas da mesma forma que a tabela 6.4. A única diferença é que instâncias que não rodaram no CPLEX devido a insuficiência de memória tem "—" em todas as suas colunas de resultado. Neste caso, a coluna de *Ótimo Relax* recebe o valor obtido através da resolução da relaxação linear do problema que sobra após a execução do algoritmo com inserção de cortes. Naturalmente, as desigualdades geradas são retiradas para a execução deste cálculo. Esses valores são apresentados para que se possa mostrar mais uma vez a efetividade das desigualdades separadas.

Nesses testes observa-se que a heurística com cortes obteve melhores limites em todos as instâncias testadas. Além disso, a otimalidade foi provada para cinco instâncias de grande porte e, quando isso não ocorreu, o *GAP* da relaxação teve uma redução bastante expressiva, fazendo com que o número de variáveis fixadas também aumentasse. A partir desses resultado, pudemos rodar o CPLEX para resolver todos os problemas no algoritmo com cortes (Tabelas 6.13 e 6.14). E novamente observamos a utilidade dos cortes na diminuição

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
80	3456	—	—	4569.36	4617	1106	99.78%	99.94%
83	3532	—	—	5208.84	5015	767	99.86%	99.96%
84	3562	—	—	6622.52	8000	623	99.90%	99.96%
85	3568	3562.7854	0.15%	6483.72	8000	11395	98.16%	99.44%
86	3620	3613.8882	0.17%	7942.72	8000	17357	97.28%	99.15%
87	3666	3656.2256	0.27%	8425.67	8000	49784	92.54%	97.70%
88	3644	—	—	6089.28	5592	646	99.91%	99.97%
89	3722	—	—	7606.61	8000	2002	99.71%	99.91%
90	3668	3661.0769	0.19%	9178.65	8000	26879	96.27%	98.91%

Tabela 6.12: Resultados obtidos com a Heurística Lagrangeana com o uso de cortes com γ=300 e 8000 iterações.

$\|P\|$	Tempo Relax	Ótimo Relax	Tempo CPLEX	Ótimo Inteiro	Iter	Nós do B&B	GAP
84	45.02	3558	337.11	3562	5522	19	0.11%
85	2299.52	3565.1429	6354.58	3568	40561	4	0.08%
86	—	3614.3333	—	3618	—	—	0.10%
87	3068.74	3660.2588	23490.31	3666	154106	51	0.16%
90	—	3660	—	3664	—	—	0.11%

Tabela 6.13: Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana sem cortes com γ=300 e 8000 iterações.

$\|P\|$	Tempo Relax	Ótimo Relax	Tempo CPLEX	Ótimo Inteiro	Iter	Nós do B&B	GAP
85	339.23	3567.3846	1662.40	3568	8767	—	0.02%
86	475.67	3617.2069	1861.57	3618	11087	—	0.02%
87	2447.77	3663.6338	7056.78	3666	31367	4	0.06%
90	955.69	3664	1010.17	3664	15694	—	—

Tabela 6.14: Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana com cortes com γ=300 e 8000 iterações.

do *GAP* de integralidade.

A estratégia de alterar parâmetros parece ser bastante interessante. No início da pesquisa, quando apenas eram usadas as instâncias de pequeno porte (15-20 pontos) o valor de γ era 40 e o limite de iterações era 1000. Esses parâmetros, quando aplicados em instâncias de médio porte, se mostraram pouco eficientes, o que nos levou a alterá-los para 80 e 2000, respectivamente. Esse indício reforça-se ainda mais com esses últimos testes.

O aumento do γ faz com que a relaxação lagrangeana consiga atingir melhores limites inferiores, mas tem o efeito colateral de demorar mais para atingir os mesmos limites e o mesmo número de variáveis fixadas do que um algoritmo com γ menor. Para indicar isto, basta observar que a execução da instância com 80 pontos através do algoritmo híbrido, com os parâmetros antigos e com uso de cortes leva menos de 3000 segundos para atingir a solução, enquanto que a mudança de parâmetros eleva esse tempo para quase 4600 segundos. A conclusão a que chegamos é que, tanto o número de iterações, quanto o valor de γ devem crescer conforme aumenta o número de variáveis do problema, e que um aumento exagerado pode fazer com que se gaste muito tempo desnecessariamente.

6.4.4 *ReStart*

A estratégia de mudar os parâmetros de execução obteve expressiva melhora quanto à qualidade das soluções obtidas para as instâncias de grande porte. Agora, vamos investigar a aplicação de uma estratégia conhecida como *ReStart*.

Essa técnica consiste em executar seguidas vezes o algoritmo lagrangeano. Antes de cada rodada os multiplicadores lagrangeanos são zerados e as desigualdades ativas são removidas, voltando à formulação original. O objetivo é aproveitar que uma grande quantidade de variáveis já foi fixada e reiniciar o processo retirando a influência dessas variáveis no valor dos multiplicadores lagrangeanos. Com isso, espera-se que o *GAP* da relaxação seja diminuído.

Para executar esses testes utilizamos os mesmos parâmetros da seção 6.4.2, ou seja, usamos geração de cortes e o valor de γ igual a 80. O número de iterações de cada execução continuou sendo de 2000 e foi permitido que o algoritmo reiniciasse no máximo 4 vezes.

Os resultados obtidos neste teste são apresentados na tabela 6.15, que segue o mesmo padrão da tabela 6.4. Comparando esses resultados com a execução original (tabela 6.8) verificamos que há uma pequena melhora no que se refere aos limites obtidos, contudo o *Restart* parece não compensar em termos de tempo. Comparando, então, com a execução que tem os parâmetros modificados (tabela 6.12) vemos que essa estratégia perde sistematicamente, tanto nos valores dos limites quanto no número de variáveis fixadas, enquanto que os tempos de execução são bastante similares.

A exceção ao que foi dito anteriormente é a instância com 80 pontos. A otimalidade

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
80	3456	—	—	2818.34	5535	356	99.93%	99.98%
83	3536	3528.27	0.22%	5615.24	10000	20594	96.36%	98.89%
84	3578	3550.33	0.77%	6268.96	10000	196042	66.98%	89.86%
85	3598	3552.60	1.26%	8126.25	10000	332693	46.26%	83.75%
86	3634	3605.44	0.78%	7437.19	10000	221759	65.21%	89.16%
87	3676	3641.08	0.95%	8500.44	10000	289485	56.62%	86.60%
88	3644	3637.62	0.18%	6093.14	10000	17118	97.49%	99.23%
89	3724	3708.61	0.41%	7690.33	10000	117070	83.25%	94.70%
90	3678	3651.00	0.73%	8735.48	10000	218707	69.67%	91.09%

Tabela 6.15: Resultados obtidos com a Heurística Lagrangeana com cortes e uso da técnica de *ReStart*.

foi provada, para esta instância, num tempo menor que o observado para os algoritmos com parâmetros modificados e com um acréscimo de tempo relativamente pequeno em relação ao original da tabela 6.8. O *ReStart* para essa instância provoca um acréscimo de aproximadamente 300 segundos, enquanto que o CPLEX demorou aproximadamente 400 para resolver o problema que sobra ao final da execução do original (tabela 6.9).

A explicação para isto é que o número de variáveis restantes após 2000 iterações da heurística lagrangeana é aproximadamente 24000, o que equivale ao número de variáveis de uma instância de 30 pontos. Conforme discutido na seção 6.4.1, para instâncias desse porte foi verificado que o método híbrido (Heurística Lagrangeana + CPLEX) já resolve melhor que o CPLEX puro. Neste caso do exemplo nem foi necessária a execução do CPLEX, já que a própria heurística lagrangeana foi capaz de provar a otimalidade.

O *ReStart* não se mostrou eficiente quando o número de variáveis restantes no problema é relativamente grande, por outro lado, quando esse número é pequeno em relação ao original esse esforço extra foi compensador. Talvez essas características possam auxiliar a minimizar o ponto fraco da mudança de parâmetros, que é justamente o fato de se gastar algum tempo para chegar a parâmetros que propiciem a resolução de uma instância específica. A idéia então seria criar um algoritmo onde os parâmetros também pudessem ser alterados na hora do *ReStart*, e além disso, chamasse o CPLEX quando restasse apenas um determinado número de variáveis. Dessa forma, seria diminuída a intervenção manual, e o processo seria mais automático.

6.4.5 Instâncias com *GAP* de Integralidade

No final da seção 6.4.2 surge uma questão sobre a inserção de planos de corte. Questiona-se se o algoritmo com cortes realmente seria mais efetivo em instâncias que apresentassem *GAP* de integralidade.

Para responder essa pergunta procuramos instâncias que tivessem essa propriedade. Usamos o CPLEX para resolver os problemas inteiros e as respectivas relaxações lineares das instâncias geradas e verificar a existência de *GAP*. Para tanto, geramos instâncias com 22 pontos, para as quais o tempo de resolução pelo CPLEX é relativamente pequeno.

Usando um gerador automático, encontramos 18 instâncias, todas elas com *GAP* absoluto entre 2 e 4 unidades e com *GAP* percentual médio de 0.22%. O algoritmo sem cortes conseguiu fechar apenas 3 destas instâncias (isso é possível devido ao método de fixação de variáveis, capítulo 5), enquanto que o algoritmo com cortes provou otimalidade em 17. A tabela 6.16 apresenta os resultados médios obtidos na execução dos dois algoritmos.

Algoritmo	$\|P\|$	GAP	Tempo	Iter	% Var Fix	% Var Fix2
Sem Cortes	22	0.207%	16.80	3750	98.48%	99.46%
Com Cortes	22	0.016%	22.24	1920	98.53%	99.48%

Tabela 6.16: Resultados médios obtidos para 18 instâncias de 22 pontos com GAP de integralidade.

Os resultados mostram que o algoritmo com planos de corte tem a possibilidade de atingir melhores limites quando há *GAP* de integralidade. E essa diferença se mostra mais clara justamente quando o limite inferior da relaxação lagrangeana chega próximo ao valor da relaxação linear. Aliando isso aos resultados apresentados na seção anterior, a nossa expectativa passou a ser de que, mudando-se os parâmetros γ e o número de iterações, o mesmo comportamento poderia ocorrer para instâncias maiores. De fato isso se verificou, pois o algoritmo com cortes conseguiu provar a otimalidade para a instância com 45 pontos. Além disso, para a instância com 58 pontos, obteve limite inferior de 3028.9431, que é melhor que o valor da relaxação linear pura (Tabela 6.6) desta instância. Essas instâncias foram escolhidas para esta análise pois, dentre aquelas testadas anteriormente, são as que já apresentavam *GAP* de integralidade.

Capítulo 7

Conclusão e Trabalhos Futuros

O trabalho apresentado neste texto pode ser dividido em duas partes: a investigação de propriedades geométricas do **RG-NLP** e a elaboração de um algoritmo exato para a sua solução.

Com relação ao primeiro tópico foram mostradas e provadas duas propriedades que permitiram a eliminação de mais de dois terços (para instâncias de grande porte) das variáveis da formulação PLI previamente proposta para o problema apenas com testes geométricos. Além disso, foi provado que, após os testes de eliminação de variáveis, um conjunto de restrições do modelo se torna combinação linear de outras restrições e, portanto, pode ser eliminado da formulação. Sendo P o conjunto de pontos que define uma determinada instância do problema, essa redução permite que $|P|$ restrições, de um total de $(|P|+1)^2$, possam ser eliminadas. A investigação das propriedades geométricas levou ainda à caracterização de duas classes de desigualdades válidas para as soluções ótimas do **RG-NLP**. Essas classes foram chamadas de desigualdades de clique e de ciclo ímpar. Foi ainda provado que o número de retângulos numa solução ótima do **RG-NLP** é fixo e igual a $|P|+1$, o que permite classificar este problema como um problema de partição de conjuntos com restrição de cardinalidade. Essa restrição se mostrou muito útil na resolução do problema através de algoritmos lagrangeanos.

O algoritmo proposto se beneficiou bastante da grande redução no número de variáveis no modelo e também das outras propriedades geométricas encontradas. Este algoritmo combina o método de relaxação lagrangeana com as técnicas de programação linear para resolver o problema, dessa maneira nós o chamamos de algoritmo híbrido.

Na primeira fase testamos duas implementações de uma heurística lagrangeana para o **RG-NLP** modelado como um problema de partição de conjuntos com restrição de cardinalidade. A primeira utilizava uma implementação tradicional de relaxação lagrangeana e a segunda um algoritmo com inserção de planos de corte automaticamente dualizados obtidos a partir das classes de desigualdades de clique e de ciclo ímpar através da resolução

de um problema de separação aplicado à solução do problema lagrangeano.

Ambas implementações se mostraram muito robustas conseguindo provar otimalidade para muitas instâncias e, quando isso não acontecia, obter bons limitantes superiores e inferiores, além de eliminar através de testes lógicos uma quantidade substancial do número de variáveis. Nos casos em que a otimalidade não era provada, utilizava-se a segunda parte do método híbrido, que é a execução de um resolvedor de PLI (CPLEX) no problema que restou após a execução da primeira fase. Com esse método conseguimos achar soluções ótimas para instâncias com até 90 pontos, quase duas vezes e meia maior do que as reportadas na literatura.

Os testes computacionais ainda mostraram que o CPLEX usado diretamente após o pré-processamento só conseguia resolver problemas com até 50 pontos devido a insuficiência de memória, além disso é possível observar que o método híbrido é mais rápido que o CPLEX puro para instâncias a partir de 30 pontos.

A comparação entre as duas implementações da relaxação lagrangeana mostra que a inserção de planos de corte foi um instrumento muito útil para instâncias grandes e para instâncias que apresentam GAP de integralidade. Outro ponto importante da pesquisa foi a investigação dos parâmetros de execução do algoritmo. Concluiu-se que o aumento do parâmetro γ (iterações sem alteração no limite inferior que devem ser executadas para mudar o multiplicador do tamanho de passo) e do número máximo de iterações têm papel fundamental na obtenção de melhores soluções, conforme aumenta o tamanho do problema.

Finalmente, ressaltamos que em nossa máquina de teste conseguimos resolver instâncias com até 90 pontos, quase duas vezes e meia as maiores instâncias resolvidas em [9]. Isso corresponde a um problema de partição com 8281 restrições e aproximadamente 720000 variáveis (2450000 antes do pré-processamento). Instâncias do problema de partição de conjuntos com este tamanho figuram entre as maiores já resolvidas até a otimalidade.

O exposto acima encoraja duas linhas de trabalho que parecem promissoras. A primeira seria a aplicação de um algoritmo híbrido como o implementado neste trabalho para a resolução de problemas genéricos de partição de conjuntos. A segunda é a exploração de técnicas de *pricing* na relaxação lagrangeana para possibilitar o aumento das instâncias resolvidas e a diminuição do seu tempo de resolução.

Apêndice A

Algoritmos de Separação de Desigualdade Válidas

Esse apêndice é uma extensão da seção 5.5. Aqui serão mostrados os algoritmos utilizados para a identificação de desigualdades violadas por uma solução do problema lagrangeano. Vale ressaltar que esses algoritmos são heurísticas e, portanto, não pretendem encontrar todo o conjunto de desigualdades violadas por uma dada solução. Desse modo, seja S a solução corrente do problema relaxado e $G = (V, E)$ seu grafo de interseção estendida associado, o algoritmo utilizado para a separação de cliques violadas é o que se segue.

1. *Para cada nó* $v \in V$
2. $\quad$ *Marca[v]* $\leftarrow$ *Falso*
3. $v \leftarrow$ *seleciona nó em* V
4. *Enquanto* $v \neq 0$ *faça*
5. $\quad$ *Marca[v]* $\leftarrow$ *Verdadeiro*
6. $\quad C \leftarrow \{v\}$
7. $\quad$ *Para cada aresta* $e = (v, v_1)$ *faça*
8. $\quad\quad$ *Para cada nó* $v_2 \in C$ *faça*
9. $\quad\quad\quad$ *Se* $(v_1, v_2) \notin E$ *então descarta* v_1
10. $\quad\quad$ *Se não descartou* v_1 *então*
11. $\quad\quad\quad C \leftarrow C \bigcup \{v_1\}$
12. $\quad\quad\quad$ *Marca[*v_1*]* $\leftarrow$ *Verdadeiro*
13. $\quad$ *Insere corte* C *no modelo*
14. $\quad v \leftarrow$ *Seleciona nó não marcado em* V

Esse algoritmo inicia em um nó arbitrário e de maneira gulosa visita sua vizinhança tentando encontrar um conjunto de nós que formem uma clique. Esse processo é repetido

UNICAMP
BIBLIOTECA CENTRAL
SEÇÃO CIRCULANTE

usando como início da procura um nó que ainda não faz parte de nenhuma clique identificada. Dessa forma a mesma clique não será identificada duas vezes na mesma iteração da heurística lagrangeana e fica permitida a participação de um dado nó em mais de uma clique. O algoritmo termina quando não existirem mais nós ainda não visitados.

Para a análise de pior caso do algoritmo vamos supor que o grafo de interseção estendida pode ser completo, já que a solução do problema relaxado pode gerar qualquer conjunto com $|P|+1$ variáveis, onde $|P|$ é a cardinalidade do conjunto de pontos da instância em questão. Isso implica que a cardinalidade do conjunto de nós é $O(|P|)$ e de arestas é $O(|P|^2)$. Nesse caso é possível ver que o algoritmo descrito acima tem tempo de execução de $O(|P|^2)$.

Utilizando o mesmo grafo foi elaborado um algoritmo para a separação de desigualdades de ciclo ímpar violadas. Esse procedimento utilizou-se do algoritmo de *busca em profundidade* [25], que é um algoritmo clássico de teoria dos grafos. Em [25] é possível verificar que esse algoritmo tem tempo de execução $O(|V| + |E|)$, o que no caso do grafo descrito acima resulta em $O(|P|^2)$. Abaixo segue em pseudo código o algoritmo utilizado para indentificação dos ciclos ímpares. Nesse caso as funções *Nível(v)* e *Pai(v)* retornam resultados calculados durante a execução da busca em profundidade. A função *Nível(v)* retorna a distância em arestas de v até o nó de início da construção da árvore de busca e *Pai(v)* retorna o nó v' adjacente a v na árvore de busca e tal que Nível(v')+1 =Nível(v).

1. $v \leftarrow$ *seleciona nó em* V
2. *Busca profundidade(v)*
3. *Para cada aresta* $e = (v, v_1)$ *faça*
4. *Se nível*(v_1) ≥ 4 *& nível*(v_1) *é par então*
5. $C \leftarrow \{v\}$
6. $v_2 \leftarrow v_1$
7. *Enquanto* $v_2 \neq v$ *faça*
8. $C \leftarrow C \bigcup \{v_2\}$
9. $v_2 \leftarrow$ *pai*(v_1)
10. *Insere corte* C *no modelo*

Esse algoritmo utiliza um nó arbitrário para montar a árvore de busca em profundidade. Essa árvore têm a importante propriedade de definir caminhos, caso eles existam, entre o nó inicial e todos os outros do grafo. Sendo assim, um nó com *Nível* par está separado por um número par de arestas do nó raiz. Caso, no grafo original, exista uma aresta ligando esses dois nós identificamos então um ciclo ímpar. A análise de pior caso deste algoritmo mostra que ele também tem tempo de execução de $O(|P|^2)$.

Apêndice B

Novos Resultados

Esse apêndice contém os resultados da execução do algoritmo híbrido para instâncias com mais de cem pontos. A máquina utilizada foi uma estação DEC/ALPHA de 675 MHz e 4GB de memória RAM rodando Linux. Essa máquina só ficou disponível para uso após a conclusão dos testes realizados no capítulo 6 e durante a própria escrita deste trabalho. Desta forma, não refizemos o estudo do efeito da mudança de parâmetros mostrado em capítulos anteriores. Apenas apresentamos esses resultados com intuito de mostrar o tempo gasto e o tamanho das instâncias tratadas quando se possui uma máquina com 4 Gigabytes de memória RAM.

As tabelas abaixo seguem o mesmo padrão das utilizadas no capítulo 6, a exceção da coluna *Tempo* nas tabelas com os resultados da heurística lagrangeana, que agora é apresentado em horas. Nas tabelas do CPLEX os tempos ainda são apresentados em segundos.

Os resultados da tabela B.1 mostram características já observadas anteriormente. A redução da densidade da matriz de restrições e o aumento percentual das reduções de pré-processamento continua ocorrendo conforme se aumenta o número de pontos da instância.

$\|P\|$	Linhas	Cols	Cols Red1	Cols Red2	% Red	Não Nulos	Dens
100	10201	3439846	3309601	1025785	29.82%	94873839	0.91%
110	12321	4657355	4494631	1364754	29.30%	138020104	0.82%
113	12996	5288643	5115389	1456847	27.55%	160680679	0.85%
115	13456	5325251	5140362	1578750	29.65%	162705240	0.77%
117	13924	6037727	5845609	1696189	28.09%	196754383	0.83%
120	14641	6317124	6116738	1819223	28.80%	210362135	0.79%
125	15876	7063529	6842731	2044619	28.95%	234747714	0.72%
130	17161	8088163	7851604	2289800	28.31%	277423076	0.71%
135	18496	8916129	8658122	2556179	28.67%	313312958	0.66%
145	21316	10903206	10604711	3188428	29.24%	405050218	0.60%

Tabela B.1: Redução de colunas. Instâncias acima de 100 pontos.

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
100	18648	18426.1484	1.19%	15.46	8000	528117	48.52%	84.65%
110	19464	19424.5625	0.20%	16.63	8000	70117	94.86%	98.49%
113	19796	19615.0840	1.81%	27.80	8000	622154	57.29%	88.24%
115	20366	20266.2031	0.49%	22.79	8000	406661	74.24%	92.36%
117	20018	19949.8203	0.34%	33.15	8000	233597	86.23%	96.13%
120	20764	20517.5078	1.19%	37.90	8000	1036175	43.04%	83.60%
125	20712	20708.3887	0.02%	32.68	8000	7474	99.63%	99.89%
130	21256	20849.6602	1.91%	58.72	8000	1733475	24.30%	78.57%
135	21906	21657.2051	1.14%	57.99	8000	1409419	44.86%	84.19%
145	22508	22312.8105	0.87%	75.86	8000	1434045	55.02%	86.85%

Tabela B.2: Resultados obtidos com a Heurística Lagrangeana sem o uso de cortes com γ=300 e 8000 iterações. Instância acima de 100 pontos

$\|P\|$	LS	LI	GAP	Tempo	Iter	Var Rest	% Var Fix	% Var Fix2
100	18482	18433.1895	0.27%	11.89	8000	98406	90.41%	97.14%
110	19452	19423.5605	0.15%	16.28	8000	45341	96.68%	99.03%
113	19706	19623.2461	0.42%	23.12	8000	268425	81.57%	94.92%
115	20332	20276.3516	0.27%	23.89	8000	220812	86.01%	95.85%
117	20032	19947.6328	0.42%	36.77	8000	330102	80.54%	94.53%
120	20746	20538.7637	1.00%	48.39	8000	923710	49.23%	85.38%
125	20712	20707.4766	0.02%	29.28	8000	3145	99.85%	99.96%
130	21074	20860.2109	1.01%	60.69	8000	1162968	49.21%	85.62%
135	21834	21666.0312	0.77%	56.72	8000	1073025	58.02%	87.97%
145	22550	22320.0508	1.02%	81.55	8000	1640719	48.54%	84.95%

Tabela B.3: Resultados obtidos com a Heurística Lagrangeana com o uso de cortes com γ=300 e 8000 iterações. Instâncias acima de 100 pontos

$\|P\|$	Tempo Relax	Ótimo Relax	Tempo CPLEX	Ótimo Inteiro	Iter	Nós do B&B	GAP
100	—	—	—	—	—	—	—
110	833.02	19438	883.23	19438	29989	0	—
125	9.28	20712	13.68	20712	2751	0	—

Tabela B.4: Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana sem cortes com γ=300 e 8000 iterações. Instâncias acima de 100 pontos.

$\|P\|$	Tempo Relax	Ótimo Relax	Tempo CPLEX	Ótimo Inteiro	Iter	Nós do B&B	GAP
100	1627.73	18454	1725.58	18454	34089	0	—
110	727.13	19438	763.68	19438	24697	0	—
125	5.78	20712	7.93	20712	1820	0	—

Tabela B.5: Resultados obtidos com o CPLEX após a execução da Heurística Lagrangeana com cortes com γ=300 e 8000 iterações. Instâncias acima de 100 pontos.

Apêndice C

Exemplos de soluções ótimas

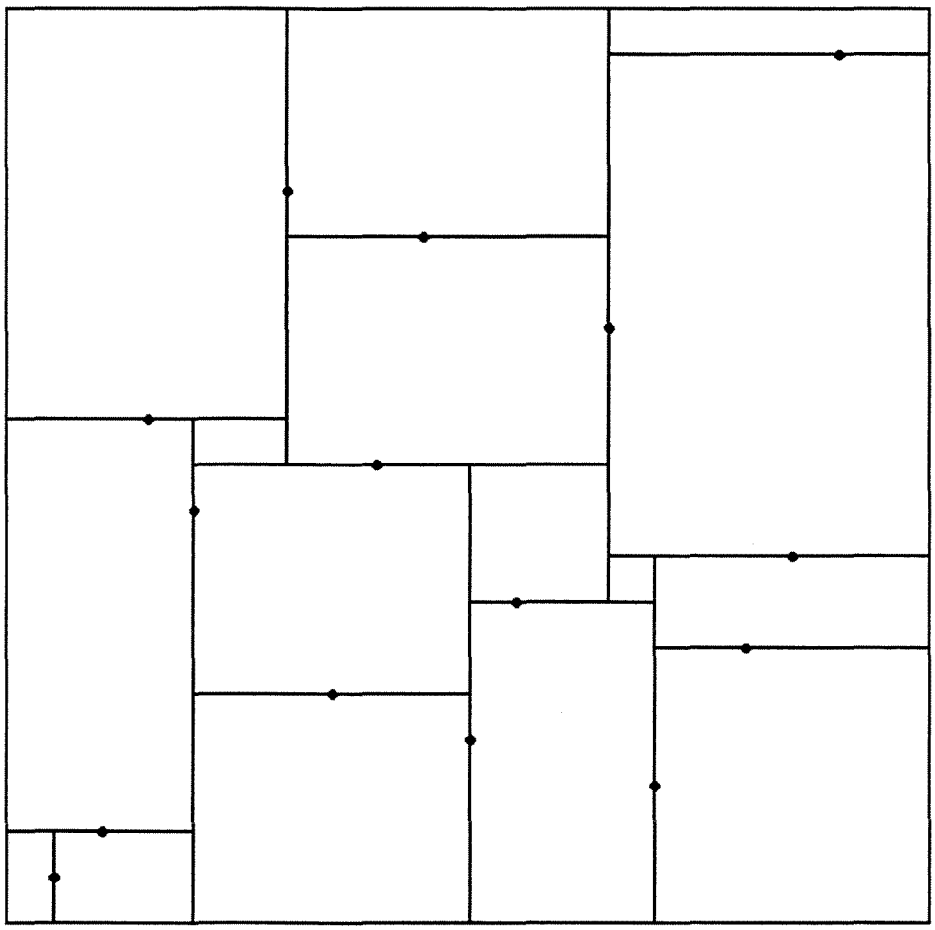


Figura C.1: Solução ótima para instância com 15 pontos.

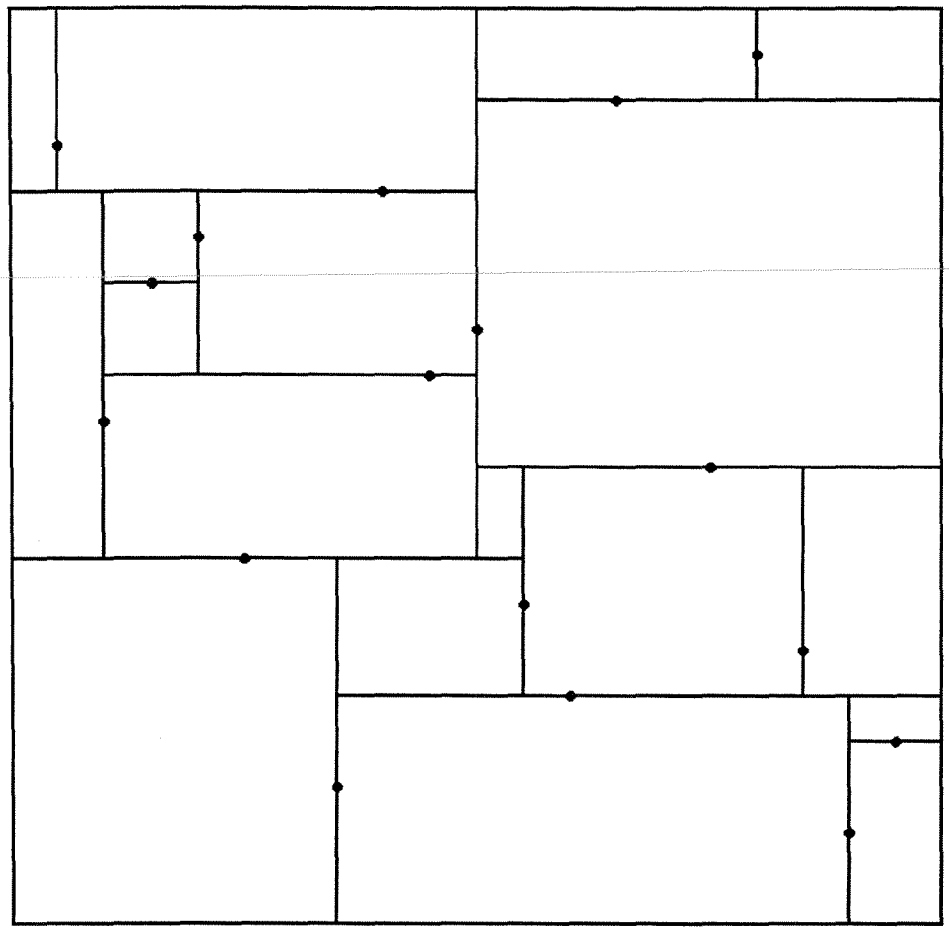


Figura C.2: Solução ótima para instância com 17 pontos.

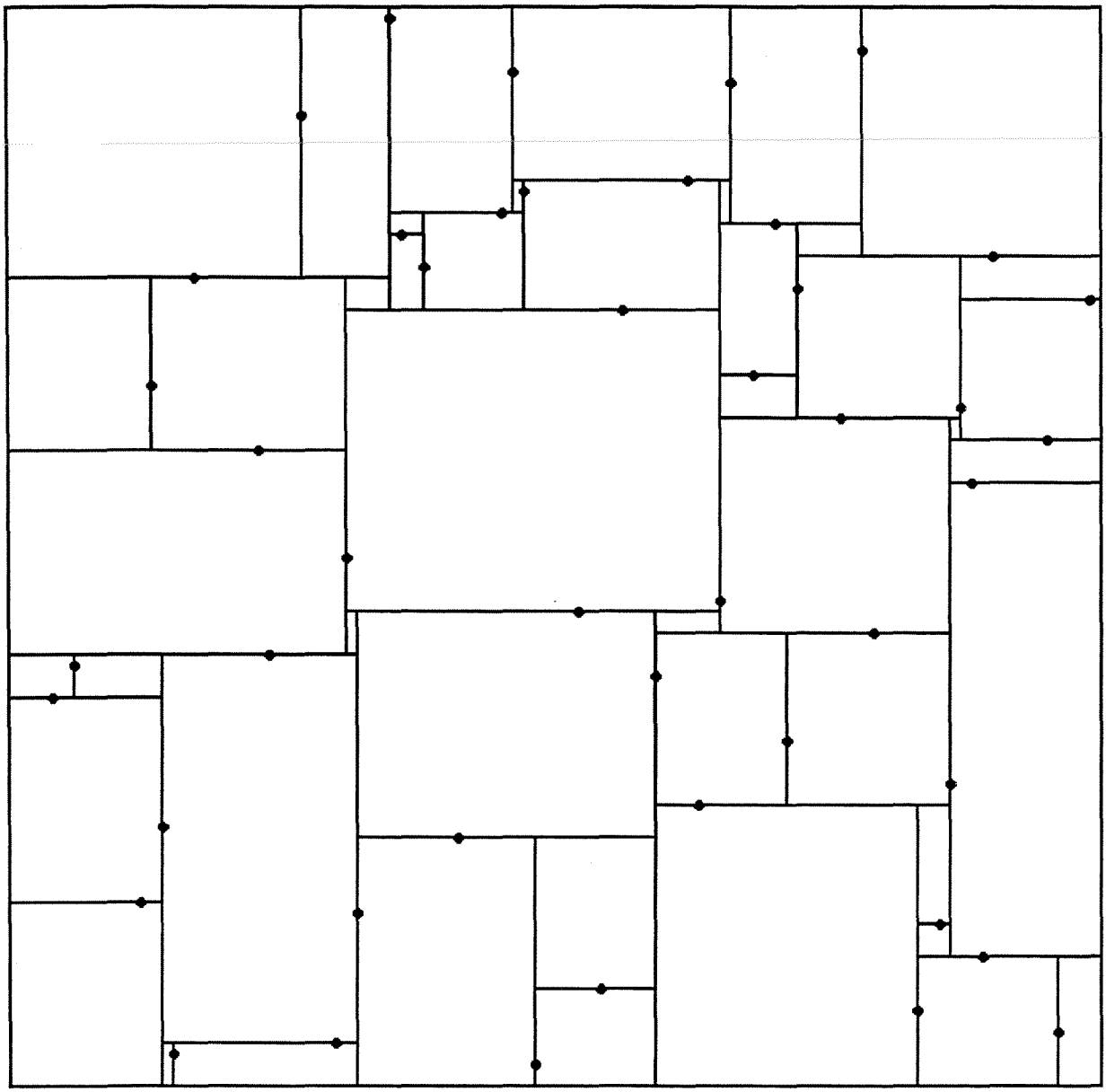


Figura C.3: Solução ótima para instância com 46 pontos.

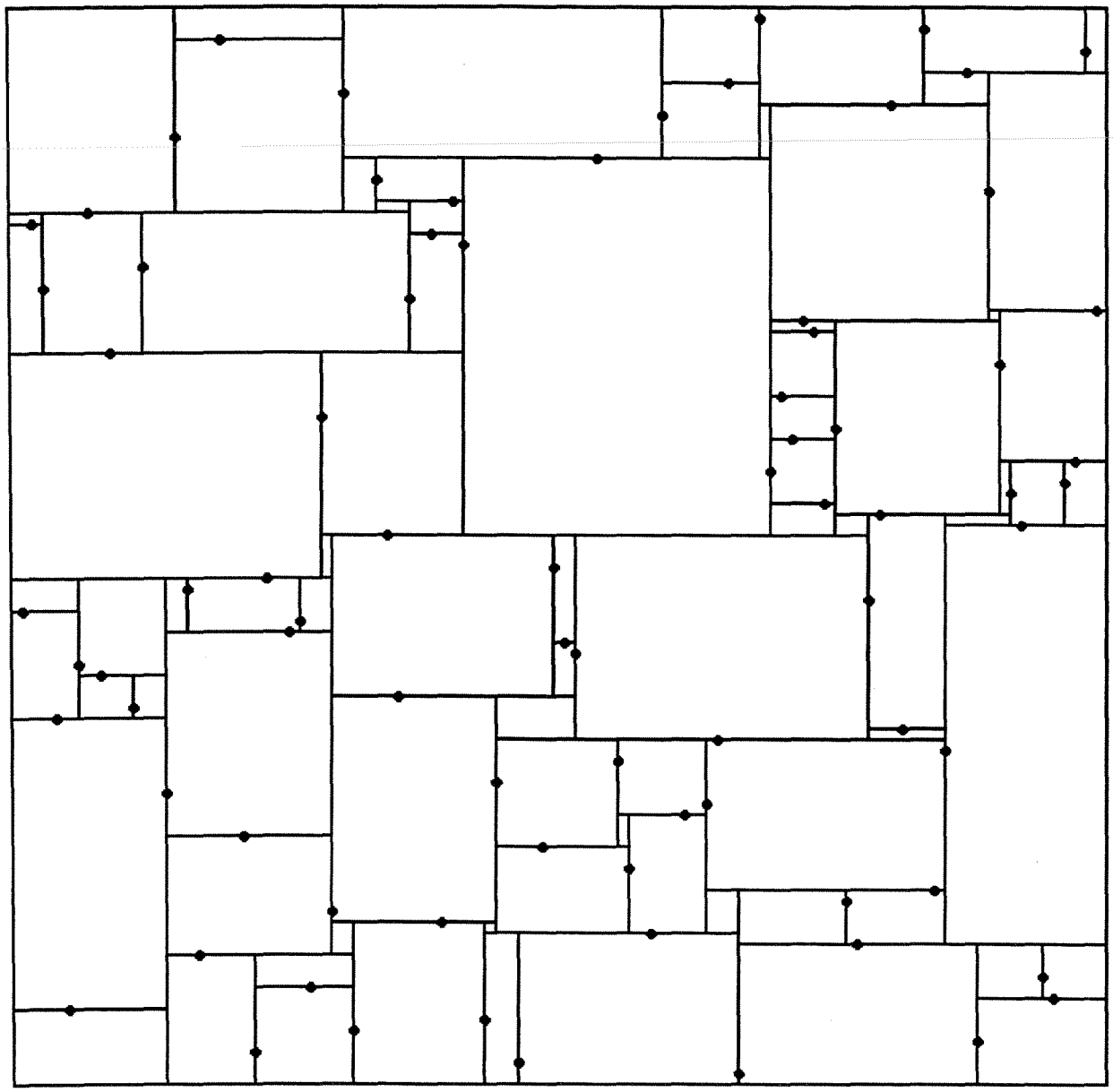


Figura C.4: Solução ótima para instância com 80 pontos.

Referências Bibliográficas

[1] A. Lingas, R. Pinter, R. Rivest and A. Shamir. Minimum Edge Length Partitioning of Rectilinear Polygons. In *Proceedings of the* 20^{th} *Annual Allerton Conference Communication, Control and Computing.* **University of Illinois, 1982.**

[2] R. Rivest. The PI (Placement and Interconnect) System. In *Proceedings of 19th Design Automation Conference*, June 1982.

[3] D. Z. Du, L. Q. Pan e M. T. Shing. Minimum Edge Length Guillotine Rectangular Partition. Technical report, **Mathematical Sciences Research Institute, University of California**, Jan 1986.

[4] T. F. Gonzalez, M. Razzazi e S. Q. Zheng. An Efficient Divide-and-Conquer Approximation Algorithm for Partitioning into d-boxes. In *International Journal of Computational Geometry and Applications*, 3(4):281-287, 1993.

[5] T. F. Gonzalez, M. Razzazi, M. T. Shing e S. Q. Zheng. On Optimal Guillotine Partitions Approximating Optimal d-boxes Partitions. In *Computational Geometry*, 4:1-11, 1994.

[6] T. F. Gonzalez e S. Q. Zheng. Bounds for Partitioning Rectilinear Polygons. In *Proceedings of Symp. Computational Geometry*, 281-287, June 1985.

[7] T. F. Gonzalez e S. Q. Zheng. Approximation Algorithms for Partitioning a Rectangle with Interior Points. In *Algorithimica*, 5:11-42, 1990.

[8] C. Levcopoulos. Fast Heuristic for Minimum Length Rectangular Partitions of Polygons. In *Proceedings of 2nd Computational Geometry Conference*, June 1986.

[9] C. N. Menezes. *Partição Retangular Mínima de um Retângulo com Pontos no Interior: Uma Abordagem em Programação Linear Inteira.* Tese de Mestrado, **Instituto de Computação, Universidade Estadual de Campinas**, 1997.

[10] C. N. Menezes e C. C. de Souza, Exact Solutions of Rectangular Partitions via Integer Programming. In *International Journal of Computational Geometry & Applications*, 10(5):477-522, 2000.

[11] M. M. Rodrigues. *Problema de Planejamento de Viagens no Transporte Coletivo.* Tese de Mestrado, **Instituto de Computação, Universidade Estadual de Campinas**, 2001.

[12] M. Held e M. R. Karp. The Traveling Salesman Problem and Minimum spannig trees. In *Operations Research*, 18:1138-1162, 1970.

[13] M. L. Fisher. The Lagrangian Relaxation Method for Solving Integer Programming Problems. In *Management Science*, 27(1):1-18, 1981.

[14] J. E. Beasley, *Lagrangean Relaxation.* **Imperial College,1992**

[15] A. Lucena. Steiner Problem in Graphs: Lagrangean Relaxation and Cutting Planes. In *COAL Bulletin*, 21:2-8, 1992.

[16] C. Martinhon, A. Lucena e N. Maculan. A Relax and Cut algorithm for the vehicle routing problem. Relatório Técnico, **Departamento de Administração, Universidade Federal do Rio de Janeiro**, 2000.

[17] L. Escudero, M. Guignard e K. Malik. A Lagrangian Relax and Cut Approach for the Sequential Ordering with Precedence Constraints. In *Annals of Operations Research*, 50:219-237, 1994.

[18] G. L. Nemhauser e G. Sigismondi. A Strong cutting plane/ branch-and-bound algorithm for node packing. In *Journal of Operational Research Society*, 43(5):443-457, 1992.

[19] M. W. Padberg. On the facial structure of set packing polyedra. In *Mathematical Programming*, 5:199-215, 1973.

[20] F. Barahona e R. Anbil. The Volume Algorithm: producing primal solutions with a subgradient method. IBM Research Report, **IBM Research division. T. J. Watson Research Center, 1997**

[21] L. A. Wolsey. *Integer Programming.* Wiley-Interscience publication, 1998.

[22] G. L. Nemhauser e L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.

[23] C. E. Ferreira e Y. Wakabayashi. *Combinatória Poliédrica e Planos-de-Corte Faciais.* X Escola de Computação, Campinas, 8-13 de Julho de 1996.

[24] R. E. Gomory. Outline of an Algorithm for integer solutions to linear programs. In *Bulletin of the American Mathematical Society* 64, 275-278, 1958.

[25] T. H. Cormen, C. E. Leiserson, e R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1998.

[26] G. Booch, J. Rumbaugh, e I. Jacobson. *The Unified Modeling Language User Guide.* Addison-Wesley Pub.

[27] CPLEX 6.5 ILOG, inc. CPLEX Division, 1999.

[28] Standard Template Library: http://www.sgi.com/technology/STL/

[29] C. E. Ferreira, J. C. De Pina, e C. G. Fernandes. Comunicação pessoal.