UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO
MESTRADO EM ENGENHARIA DE PRODUÇÃO

MARCOS COSTA ROBOREDO

SUAVIZANDO A FUNÇÃO DUAL LAGRANGEANA QUANDO O SUBPROBLEMA É RESOLVIDO POR PROGRAMAÇÃO DINÂMICA

NITERÓI
2009

MARCOS COSTA ROBOREDO

SUAVIZANDO A FUNÇÃO DUAL LAGRANGEANA QUANDO O SUBPROBLEMA É RESOLVIDO POR PROGRAMAÇÃO DINÂMICA

Dissertação apresentada ao Curso de Pós-Gradução em Engenharia de Produção da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Sistemas, Apoio à decisão e Logística.

Orientador: Prof. Dr. ARTUR ALVES PESSOA

Niterói

2009

MARCOS COSTA ROBOREDO

SUAVIZANDO A FUNÇÃO DUAL LAGRANGEANA QUANDO O SUBPROBLEMA É RESOLVIDO POR PROGRAMAÇÃO DINÂMICA

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia de Produção da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Sistemas, Apoio à decisão e Logística.

Aprovada em Dezembro de 2009.

BANCA EXAMINADORA

Prof. Dr. ARTUR ALVES PESSOA - Orientador
UFF

Prof. Dr. EDUARDO UCHOA BARBOZA
UFF

Prof. Dr^{a} LAURA SILVIA BAHIENSE DA SILVA LEITE
UFRJ

NITERÓI
2009

Dedico este trabalho à minha família e à minha namorada Luana.

Agradecimentos

A Deus.

A meus pais, Carlos e Claudia e aos meus irmãos Rodrigo, Gustavo e Bruno por terem acreditado e torcido por mim.

À minha namorada Luana pelo apoio incondicional mesmo nos meus momentos de ausência.

Ao professor Artur Alves Pessoa pelo apoio, por ter me orientado de maneira efetiva neste trabalho e pela atenção a mim dedicada em todos as etapas deste trabalho.

A todos professores do Mestrado.

Aos amigos do LOGIS, Luiz, Hugo e José Maurício pelo apoio e pelas boas idéias dadas.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Ao utilizar relaxações lagrangeanas para resolver problemas de otimização combinatória, a função dual lagrangeana resultante não é diferenciável, o que pode gerar grandes dificuldades de convergência. Neste trabalho, introduzimos uma nova técnica de suavização para funções duais lagrangeanas associadas a problemas de otimização combinatória quando o subproblema é resolvido por programação dinâmica. Nossa técnica permite a utilização de métodos de gradiente de subida para aproximar os valores ótimos dos multiplicadores de Lagrange para o problema original. Também provamos que, quando o gradiente obtido é zero, uma solução fracionária viável para o problema original é dada. Utilizamos esta técnica para calcular limites inferiores para o Problema de Atribuição Generalizado. Para este problema, nossos experimentos mostram que limites inferiores muito próximos aos ótimos podem ser obtidos em um tempo computacional modesto quando comparado ao melhor algoritmo de geração de colunas estabilizada encontrado na literatura.
Palavras-Chave: Otimização. Relaxação Lagrangeana. Métodos do gradiente. Programação Dinâmica.

Abstract

When lagrangean relaxation is applied to combinatorial optimization problems, the resulting dual function becomes non-differentiable, which may generate serious convergence difficulties. In this paper, we introduce a new technique for smoothing the lagrangean dual function associated to relaxed combinatorial optimization problems when the subproblem is solved via dynamic programming. Our technique allows for using gradient ascent methods to approach the optimal values of the lagrangean multipliers for the original problem. We also prove that when the obtained gradient is zero, a feasible fractional solution for the original problem is given. We use this technique to calculate lower bounds for the classical Generalized Assignment Problem. For this problem, our experiments show that lower bounds very close to the optimal ones can be found in a modest computational time, when compared to the best stabilized column generation algorithm found in the literature.

Keywords: Optimization. Lagrangean Relaxation. Gradient methods. Dynamic programming.

Capítulo 1

Introdução

Ao lidar com problemas de otimização, em muitos casos, podemos observar que a dificuldade é amplamente reduzida se retirarmos um conjunto relativamente pequeno de restrições. Nesse tipo de caso, é indicado o método da Relaxação Lagrangeana. Ao se aplicar tal técnica são gerados um subproblema e uma função a ser otimizada denominados subproblema lagrangeano e função dual lagrangeana respectivamente.

Em problemas de otimização combinatória, a função dual lagrangeana não é diferenciável, e em nuitos casos, o subproblema é resolvido através de programação dinâmica (Bellman, 1966). Dentre esses problemas destacamos: O PAG (Problema da Alocação Generalizada) (Pigatti et al., 2005), Escalonamento em uma máquina minimizando o atraso total ponderado ($1||\sum w_j T_j$) (Rodrigues et al., 2008), PAQG (Problema da Alocação Quadrática Generalizado) (Pessoa et al., 2008), etc. Quando a função dual lagrangeana não é diferenciável, nós não podemos aplicar métodos como Método Gradiente (Hestenes, 1969), Método gradiente conjugado (Hestenes e Stiefel, 1952), dentre outros.

Neste trabalho, introduzimos uma nova técnica para suavizar a função dual lagrangeana associada a problemas de otimização combinatória relaxados quando o subproblema é resolvido através de programação dinâmica. A função suavizada nesse caso é côncava, diferenciável em todo o seu domínio e se aproxima inferiormente da função original controlada por um parâmetro.

Na literatura, podemos encontrar alguns trabalhos que também propõem a suavização de funções não-lineares em problemas de otimização (ver por exemplo Moreno et al. (2008) e Di Bello (2005)).

Como consequência para o nosso contexto, a técnica de suavização da função dual

lagrangeana permite o uso de métodos gradientes para a aproximar os valores ótimos dos multiplicadores lagrangeanos para o problema original. O gradiente da nossa função suave é calculado através de um algoritmo de programação dinâmica modificado. Durante esse cálculo, o gradiente também obtém uma solução fracionária que pode ser escrita como uma combinação convexa de soluções viáveis para o subproblema original. Nós provamos que o gradiente obtido é igual ao vetor das violações das restrições para esta solução fracionária. Como resultado, encontrando um vetor de multiplicadores lagrangeanos que leva ao gradiente zero leva também a uma solução viável fracionária para o problema original. Com o objetivo de simplificar, nós apresentamos nossa técnica, teoria e resultados experimentais no contexto do PAG. Para este problema, nossos experimentos mostram que limites muito próximos do limite ótimo podem ser encontrados em modestos tempos computacionais, quando comparado ao melhor algoritmo de geração de colunas estabilizda encontrado na literatura. Esta dissertação está organizada da seguinte maneira: nos capítulos 2, 3, 4 e 5 é feita uma revisão de literatura, conceitos e resultados fundamentais no contexto da Relaxação Lagrangeana, Programação Dinâmica, Geração de Colunas e Métodos clássicos de subida, respectivamente. No capítulo 6 é apresentado e definido o PAG juntamente com uma possível relaxação para este problema. O capítulo 7 mostra a suavização para a função dual lagrangeana da relaxação do PAG enquanto o capítulo 8 mostra como é feito o cálculo do gradiente. O método Quase-Newton de subida para ajuste dos multiplicadores de lagrange é detalhado no capítulo 9. Os resultados computacionais, bem como conclusões a respeito destes, estão no capítulo 10. As conclusões e perspectivas de trabalhos futuros são apresentadas no capítulo 11.

Capítulo 2

Relaxação Lagrangeana

Como já comentado de maneira sucinta na introdução, diversos problemas de otimização combinatória, difíceis de serem resolvidos, podem ter sua dificuldade amplamente reduzida se não consideramos, no problema, um número relativamente pequeno de restrições. Para este tipo de caso, podemos aplicar o método da Relaxação Lagrangeana (Fisher, 1985), pois este dualiza as restrições que dificultam o problema, ou seja, as adiciona à função objetivo, penalizando-as através de um vetor de multiplicadores denominados multiplicadores de Lagrange. O problema gerado é denominado subproblema lagrangeano e a solução ótima para este é um limite dual para o problema original. A técnica que consiste em resolver uma seqüência de subproblemas variando os valores dos multiplicadores de lagrange é denominada Relaxação Lagrangeana. A cada resolução de um subproblema, um novo limite para o problema original é gerado. A função que associa cada vetor de valores dos multiplicadores ao correspondente limite é chamada função dual lagrangeana.

Held e Karp (1971) usaram pela primeira vez o método da Relaxação Lagrangeana para o problema do caixeiro viajante (Lawler, 1985), sendo o nome Relaxação Lagrangeana usado pela primeira vez por Geoffrion (1974).

Qualquer problema de otimização combinatória com restrições pode ser submetido ao método da Relaxação Lagrangeana. Assim, no contexto da Engenharia de Produção podemos encontrar na literatura diversos problemas que foram submetidos a este método. Dentre estres, podemos destacar: problemas de roteamento como o problema do caixeiro viajante, problemas de *scheduling* como por exemplo escalonamento em uma máquina minimizando o atraso total ponderado, problemas de atribuição como o PAG e o PAQG. A tabela 2.1 mostra algumas destas aplicações juntamente com uma referência e o sub-

problema lagrangeano associado.

Tabela 2.1: Aplicações da relaxação lagrangeana

Problema	Pesquisadores	Subproblema Lagrageano
Atribuição		
Alocação generalizada (PAG)	Fisher et al. (1986)	Mochila booleana
Alocação quadrática generalizado (PAQG)	Pessoa et al. (2008)	Programação dinâmica
Roteamento		
Caixeiro viajante (PCV)	Held e Karp (1971)	Árvore geradora
Roteamento de veículos com janelas de tempo	Kohl e Madsen (1997)	Caminho mínimo com tempo e capacidade
Scheduling		
1 máquina minimizando o atraso total ponderado	Fisher (1976)	Programação dinâmica pseudo-polinomial
n máquinas minimizando o atraso total ponderado	Fisher (1973)	Programação dinâmica pseudo-polinomial

Baseado em Guignard (2003), a relaxação lagrangeana é definida como segue.

Considere o seguinte problema de otimização:

$$\min_x \{\, f(x) \mid Ax \leq b,\, Cx \leq d,\, x \in X \,\}, \qquad [P]$$

onde X contém possíveis restrições de sinal sobre x, restrições de integralidade e etc. Assumimos que as restrições $Ax \leq b$ dificultam o problema $[P]$ no sentido de que sem elas, o problema se torne bem mais simples para resolver e portanto serão dualizadas. Seja θ um vetor de números não-negativos de pesos para representar os multiplicadores de Lagrange.

A Relaxação Lagrangeana para o problema $[P]$, que dualiza as restrições $Ax \leq b$, com multiplicadores θ é o problema

$$\min_x \{\, f(x) + \theta(Ax - b) \mid Cx \leq d,\, x \in X \,\}. \qquad [L(\theta)]$$

$[L(\theta)]$ é chamado de *subproblema lagrangeano*. Notemos que as restrições $Ax \leq b$ estão sendo adicionadas a função objetivo e cada uma delas penalizada por uma componente do vetor de pesos θ, ou seja, estão sendo dualizadas. $[L(\theta)]$ é uma relaxação de $[P]$.

O próximo teorema mostra duas propriedades válidas para a relaxação lagrangeana, onde $FS(Y)$ denota o conjunto das soluções viáveis para o problema Y.

Teorema 1 *Dados a relaxação lagrangeana* $[L(\theta)]$ *e o problema* $[P]$ *já definidos, as seguintes propriedades são válidas:*

- $FS(P) \subset FS(L(\theta))$
- *Para qualquer* x *viável de* $[P]$*, e qualquer* $\theta > 0$*,* $f(x) + \theta(Ax - b) \leq f(x)$*. Isto mostra que* $L(\theta) \leq P$*, para todo* $\theta \geq 0$*, i.e., o ótimo valor* $L(\theta)$*, que depende de* θ*, é um limite inferior para o valor ótimo de* $[P]$*.*

Prova: Para provar a 1ª propriedade, devemos mostrar que dada uma solução viável de $[P]$ esta solução também é viável para $[L(\theta)]$. Para isso, seja $x_1 \in X$ solução viável para $[P]$. Assim, x_1 satisfaz os seguintes grupos de restrições: $Ax \leq$ b e $Cx \leq$ d logo se x_1 satisfaz $Cx \leq d$ então x_1 é solução viável para $[L(\theta)]$.
Sejam x_2 solução viável para $[P]$ e θ_1 um vetor de multiplicadores de lagrange qualquer, para provar a 2ª propriedade, devemos mostrar que:

$$f(x_2) + \theta_1(Ax_2 - b) \quad \leq \quad f(x_2)$$

A inequação acima é válida pois $\theta_1 \geq 0$ e como x_2 é viável para $[P]$ temos $Ax_2 - b \leq 0$ logo $\theta_1(Ax_2 - b) \leq 0$. ■

A segunda propriedade do teorema 1 é chamada de *dualidade fraca*.

O próximo teorema mostra uma nova propriedade no contexto da relaxação lagrangeana. Tal propriedade é chamada *dualidade forte*.

Teorema 2 *Dado* $\bar{x}$ *solução ótima do subproblema lagrangeano para* θ*. Se* $\bar{x}$ *é viável para* $[P]$ *e* $\theta(Ax - b) = 0$ *então* $L(\theta) = P$ *para* $\bar{x}$*.*

Prova: Basta notarmos que pelo fato de $\bar{x}$ ser viável para $[P]$ podemos calcular P para $\bar{x}$ e notarmos que se $\theta(Ax - b) = 0$ então $A\bar{x} - b = 0$ sempre que $\theta_i \geq 0$. Assim,
$L(\theta) = f(\bar{x}) + \theta(A\bar{x} - b) = f(\bar{x}) = P$ ■

O problema de encontrar o limite inferior mais próximo de P, isto é:

$$max \quad L(\theta)$$
$$sujeito\,a \quad \theta \geq 0$$

é chamado de *problema dual lagrangeano* do problema $[P]$ relativo as restrições dualizadas $Ax \leq b$ e a função que associa cada vetor de multiplicadores θ à $L(\theta)$ é chamada de *função dual lagrangeana*.

Suponhamos agora que as restrições que dificultam o problema são de igualdade. Assim, por exemplo, ao invés de $Ax \leq b$ teríamos $Ax = b$. Nesse caso, basta percebermos que o grupo de restrições $Ax = b$ pode ser substituido por $Ax \leq b$ e $-Ax \leq -b$ e dualizarmos esses dois grupos de restrições através de dois vetores de multiplicadores. Ao invés disso, utilizamos apenas um vetor de multiplicadores π para restrições de igualdade, onde cada componente de π pode ser negativa.

Com intuito de facilitar a notação representamos a função objetivo do subproblema lagrangeano em função do vetor de multiplicadores de lagrange θ e π e da variável x por $L(x, \theta, \pi)$ e a função dual em função desses mesmos vetores por $L(\theta, \pi)$, onde θ está associado a restrições de desigualdade e π a restrições de igualdade.

Com intuito de ilustrar e facilitar o entendimento do conceito de Relaxação Lagrangeana, é proposto um pequeno problema de otimização. Este problema possui somente duas possíveis soluções: s_1 e s_2. cada uma das duas soluções tem um custo e faz uso de unidades de um determinado recurso, onde custo$(s_1) = 2$, custo$(s_2) = 10$, uso(s_1)= 35, e uso$(s_2) = 15$. Nosso objetivo é minimizar o custo, com a restrição de que o uso do recurso não exceda 20 unidades. A formulação do problema é:

$$\begin{aligned} \text{Minimizar} \quad & \text{custo}(x) \\ \text{sujeito a} \quad & \text{uso}(x) \leq 20 \\ & x \in \{s_1, s_2\} \end{aligned} \tag{2.1}$$

Notemos que a solução s_1 não é viável pois o uso(s_1)= 35 viola a restrição 2.1. Assim, a solução para problema prévio é s_2 (s_2 não viola 2.1), gerando um custo ótimo custo$(s_2) = 10$. Nós aplicamos o método da Relaxação Lagrangeana para este problema dualizando a restrição(2.1). Seja θ o multiplicador lagrangeano correspondente. O subproblema lagrangeano torna-se:

$$\begin{aligned} \text{Minimizar} \quad & L(x, \theta) = \text{custo}(x) + \theta(\text{uso}(x) - 20) \\ \text{sujeito a} \quad & x \in \{s_1, s_2\} \end{aligned}$$

Notemos que a restrição (2.1) foi adicionada à função objetivo penalizada pelo multiplicador de lagrange θ.

O problema dual lagrangeano é:

$$\begin{aligned} &\text{Maximizar} \quad L(\theta) \\ &\text{Sujeito a} \quad \theta \geq 0. \end{aligned} \tag{2.2}$$

Onde $L(\theta)$ representa a função dual lagrangeana que associa o multiplicador θ ao limite inferior para o problema.

Se x^* é uma solução ótima do subproblema lagrangeano, a função dual lagrangeana é:

$$L(\theta) \quad = \quad L(x^*, \theta). \tag{2.3}$$

É fácil notarmos que:

- $L(s_1, \theta) = 2 + 15\theta$
- $L(s_2, \theta) = 10 - 5\theta$
- $L(\theta) = min\{2 + 15\theta, 10 - 5\theta\}$.

A figura 2.1 mostra o gráfico da função dual lagrangeana. Note, por exemplo, que $L(\theta = 0,2) = 5$ é um limite inferior para o custo ótimo do problema que é 10. Note também que $L(\theta = 0,4) = 8$ representa o máximo da função dual lagrangeana, ou seja, o melhor limite inferior para o problema nesse caso.

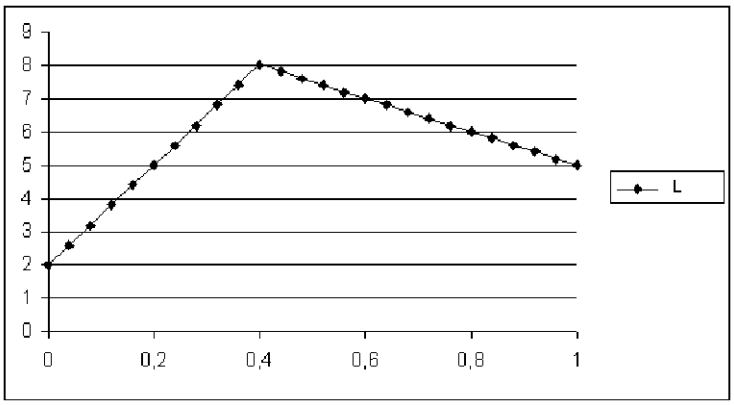

Figura 2.1: A função dual lagrangeana para o problema

Notemos que a função dual lagrangeana $L(\theta)$ nesse caso não é derivável somente devido à função *mínimo* (o mesmo acontece para o problema estudado nessa dissertação) e portanto para maximizá-la não podemos usar métodos como gradiente, gradiente conjugado, dentre outros.

Uma importante propriedade das funções côncavas é que estas adimitem supergradientes, enquanto as funções convexas admitem subgradientes. Todos os resultados referentes a supergradientes e subgradientes são análogos.

Um supergradiente de $L(\theta, \pi)$ no ponto (θ^0,π^0) é qualquer vetor $\binom{\alpha}{\beta}$ tal que:

$$L(\theta, \pi) \leq L(\theta^0, \pi^0) + \alpha^\intercal(\theta - \theta^0) + \beta^\intercal(\pi - \pi^0). \tag{2.4}$$

A interpretação geométrica para o supergradiente é que este é um hiperplano suporte para a função dual tangente ao ponto considerado.

O próximo teorema mostra um importante resultado quando a função dual possui supergradiente zero em algum ponto do seu domíno.

Teorema 3 *Se $L(\theta, \pi)$ tem supergradiente zero no ponto (θ^0, π^0) então este ponto é ótimo para o problema dual lagrangeano.*

Prova: Como $L(\theta, \pi)$ tem supergradiente zero no ponto (θ^0, π^0) então

$$\begin{aligned} L(\theta, \pi) &\leq L(\theta^0, \pi^0) + 0(\theta - \theta^0) + 0(\pi - \pi^0) && (2.5)\\ &= L(\theta^0, \pi^0). && (2.6) \end{aligned}$$

Assim, $L(\theta^0, \pi^0)$ é o máximo da função dual lagrangeana. ■

Teorema 4 *Se $\bar{x}$ é solução ótima do subproblema lagrangeano no ponto (θ^0, π^0) então $(g(\bar{x}), h(\bar{x}))$ é um supergradiente de $L(\theta, \pi)$ neste ponto, onde $g(\bar{x})$ e $h(\bar{x})$ são vetores que representam a violação das restrições associadas a θ e π respectivamente .*

Prova: Observemos primeiramente que: $L(\theta^0, \pi^0) = L(\bar{x}, \theta^0, \pi^0) = f_{\bar{x}} + \theta^{0\intercal} g(\bar{x}) + \pi^{0\intercal} h(\bar{x})$. Esta igualdade é usada na inequação abaixo para quaisquer θ e π.

$$
\begin{aligned}
L(\theta,\pi) &\leq L(\bar{x},\theta,\pi) && (2.7)\\
&= f_{\bar{x}} + \theta^{\intercal} g(\bar{x}) + \pi^{\intercal} h(\bar{x}) && (2.8)\\
&= f_{\bar{x}} + g(\bar{x})^{\intercal}(\theta^0 + \theta - \theta^0) + h(\bar{x})^{\intercal}(\pi^0 + \pi - \pi^0) && (2.9)\\
&= f_{\bar{x}} + \theta^{0\intercal} g(\bar{x}) + \pi^{0\intercal} h(\bar{x}) + g(\bar{x})^{\intercal}(\theta - \theta^0) + h(\bar{x})^{\intercal}(\pi - \pi^0) && (2.10)\\
&= L(\theta^0,\pi^0) + g(\bar{x})^{\intercal}(\theta - \theta^0) + h(\bar{x})^{\intercal}(\pi - \pi^0) \quad \blacksquare && (2.11)
\end{aligned}
$$

No exemplo dado, nenhuma solução viável para o subproblema fornece um supergradiente zero.

Em geral, métodos de busca por supergradientes (ou subgradiente para o caso de minimização) não pussem bons critérios de parada, apesar de serem bastante usados. Tipicamente, o método é parado antes de se encontrar o limite ótimo.

Pelo fato do método da Relaxação Lagrangeana gerar bons limites para diversos problemas de otimização combinatória, este método tem sido aplicado em conjunto com o algoritmo de resolução exata Branch and Bound (Lawler e Wood, 1966), valendo-se dos bons limites gerados pelas relaxações, com intuito de diminuir o tempo de resolução do problema e o número de iterações na *árvore de Branch and Bound*. Um exemplo de um estudo feito utilizando Relaxação Lagrangeana como gerador de limites para o algoritmo de Branch and Boundpode ser visto em Volgenant e Jonker (1982).

Ao aplicarmos o método da Relaxação Lagrangeana, desejamos encontrar: bons multiplicadores (multiplicadores que levem a limites próximos ou até mesmo iguais ao limite ótimo), em tempos razoáveis, e em um número relativamente pequeno de iterações. Tais desejos têm gerado diversos estudos com relação a métodos de ajuste dos multiplicadores(MAM)(ver Erlenkotter (1978) e Galvao et al. (2000)).

Segundo Espejo e Galvão (2002), os problemas duais da Relaxação Lagrangeana são resolvidos, em geral, através de métodos supergradiente (subgradiente se o problema dual é de minimização), como por exemplo em Karabakal et al. (1992), e os algoritmos utilizados para resolver o problema dual lagrangeano são, basicamente, o método dos subgradientes, variantes dos procedimentos de decomposição, sejam eles de Benders (Schrijver, 1998) ou Dantzig-Wolfe (Barahona e Anbil, 2000).

Capítulo 3

Programação Dinâmica e o Problema da Mochila

Programação dinâmica é um algoritmo amplamente utilizado para resolução de alguns tipos de problemas de otimização combinatória. A estratégia da programação dinâmica é dividir o problema em subproblemas independentes e resolvê-los recursivamente, guardando as soluções já calculadas com o intuito de evitar de ter que recuperá-las. Alguns problemas de otimização combinatória que podem ser resolvidos via programação dinâmica são:

- Problema da mochila (KP)
- Problema da parentização ótima de multiplicação de matrizes (Cormen et al., 2001)
- Problema de escalonamento minimizando o atraso total (Pinedo, 2008)

Dentre os problemas acima, o que nos interessa é o problema da mochila (*Knapsack Problem*) pois este é subproblema do problema estudado nessa dissertação (o PAG). O problema da mochila é um clássico problema de otimização combinatória amplamente , pois, além de possuir diversas aplicações práticas, aparece como subproblema ou subestrutura de diversos outros problemas. O problema da mochila é definido da seguinte maneira:

Dada uma mochila de capacidade $W \in \mathbb{N}$, um conjunto finito de itens $I = \{1, ..., n\}$, cada item $i \in I$ possui um peso w_i e um lucro p_i que também são dados. O objetivo do problema é encontrar o subconjunto de itens que maximize o lucro e respeite a capacidade da mochila.

Existem algumas variantes para o problema da mochila tais como:

- Problema da mochila fracionário;
- Problema da mochila booleano (*0-1 knapsack problem*);
- Problema da mochila limitado (*Bounded knapsack problem*);
- Problema da mochila não limitado (*Unbounded knapsack problem, UKP*).

O problema da mochila fracionário permite que frações dos itens sejam colocadas na mochila, o problema da mochila booleano considera a seguinte decisão: ou um determinado item é colocado na mochila ou não é colocado, o problema da mochila limitado possui limitação para o número de vezes que um determinado tipo de item entre na mochila enquanto o problema da mochila ilimitado não possui essa limitação.

Dentre as variações para o problema da mochila, estamos interessados pelo problema da mochila booleano (*0-1 knapsack problem*). Uma formulação para o problema consiste em considerar a variável x_i, onde $x_i = 1$ se o item i é colocado na mochila e $x_i = 0$ caso contrário. Assim, temos:

$$
\begin{aligned}
& [KP] & & \\
& \max \quad \sum_{i=1}^{n} p_i x_i & & (3.1)\\
& \text{sujeito a} \quad \sum_{i=1}^{n} w_i x_i \leq W \quad i \in I & & (3.2)\\
& \qquad x_i \in \{0,1\} \quad i \in I. & & (3.3)
\end{aligned}
$$

Para resolver o problema $[KP]$ podemos usar programação dinâmica. Para isso, consideremos a matriz OPT, com células $OPT(l,k)$, onde $l \in \{0,1,...,n\}$ e $k \in \{0,1,...,W\}$. Se $l = 0$, Consideremos $OPT(l,k) = 0$ e caso contrário $OPT(l,k)$ representa o custo da solução considerando $\{1,...,l\}$ como conjunto de itens e k como capacidade da mochila. A idéia para montar esta matriz através de uma recursão é a seguinte: se a mochila ótima para os itens $\{1,...,l\}$ não usa o item l então ela é também uma mochila ótima para $\{1,...,l-1\}$. Já se a mochila ótima para $\{1,...,l\}$ e usa o item l então ela é o resultado de juntar o item l com uma mochila ótima para $\{1,...,l-1\}$. Matematicamente temos:

$$
OPT(l,k) = \begin{cases} 0 & \text{se} \quad l = 0 \quad \text{ou} \quad k = 0. \\ OPT(l-1,k) & \text{se} \quad l \neq 0 \quad \text{e} \quad k < w_l \\ max\{OPT(l-1,k), OPT(l-1,k-w_l) + p_l\} & \text{se} \quad l \neq 0 \quad \text{e} \quad k \geq w_l. \end{cases} \tag{3.4}
$$

Como estamos interessados na solução que envolve todos os n itens do problema e a capacidade da mochila W, o custo da solução ótima é fornecido pela célula $OPT(n, W)$.

Para descobrirmos o valor das variáveis binárias x_i, com $i \in \{1, ..., n\}$, usamos o seguinte algoritmo:

Algoritmo 1 Determinando o valor do vetor x

$Y \leftarrow W$
para $j = n$ **até** 1 **faça**
  se $OPT[j, Y] = OPT[j-1, Y]$ **então**
    $x_j \leftarrow 0$
  senão
    $x_j \leftarrow 1$
    $Y \leftarrow Y - w_j$
  fim se
fim para

Consideremos uma pequena instância para o problema da mochila com 4 itens, $W = 6$ e onde o peso e o lucro de cada item são dados na tabela abaixo:

Item	1	2	3	4
Peso	2	3	1	2
Lucro	100	300	200	250

A matriz OPT que fornece o custo da solução do problema é preenchida através da recursão mostrada em (3.4):

$$OPT(l, k) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 100 & 100 & 100 & 100 \\ 0 & 0 & 100 & 300 & 300 & 400 & 400 \\ 0 & 200 & 300 & 300 & 500 & 500 & 600 \\ 0 & 200 & 250 & 450 & 450 & 550 & 750 \end{pmatrix}$$

O custo da solução é $OPT(4, 6) = 750$. Através do algoritmo 1 obtemos os valores das variáveis x_1, x_2, x_3 e x_4 que estão mostrados na tabela abaixo:

Variável	Valor
x_1	0
x_2	1
x_3	1
x_4	1

Como podemos notar, apenas o item 1 não foi colocado na mochila. Esta solução é uma solução viável pois o peso colocado na mochila é $w_2 + w_3 + w_4 = 6$ e não excede a sua capacidade $W = 6$. O custo de 750 dado por essa solução está correto pois $p_2 + p_3 + p_4 = 750$.

É fácil percebermos que o algoritmo de programação dinâmica para resolução do problema da mochila é resolvido em tempo $\Theta(nW)$ onde n é o número de itens e o W é a capacidade da mochila. Este tempo de execução é devido ao preenchimento da matriz recursiva OPT. Podemos dizer que o tempo de execução deste algoritmo é exponencial uma vez que

$$\Theta(nW) \quad = \quad \Theta(n2^{\log_2 W}),$$

onde $\log_2 W$ é o número de bits necessários para representar o número W. Por exemplo, imaginemos uma mudança na escala de pesos em que os pesos e preços dos itens $w_1, ..., w_n, p_1, ..., p_n$ e a capacidade da mochila W sejam multiplicados por 6000. Assim, teríamos o mesmo problema porém o tempo de consumo 6000 vezes maior para resolvê-lo.

Por este motivo, o algoritmo de programção dinâmica para o problema da mochila é dito *pseudo-polinomial* pois é polinomial sobre o valor de W e não sobre seu número de bits.

Capítulo 4

Simplex revisado, decomposição de Dantzig-Wolfe e Geração de Colunas

O Simplex revisado, como o próprio nome sugere, é uma revisão para o método Simplex, que é um conhecido método de resolução de problemas de programação linear. Basicamente, a versão clássica do Simplex armazena diversas variáveis desnecessárias, o que faz com que o tempo de resolução do problema aumente. Para aplicação do método do simplex revisado fazem-se necessários 6 passos. O 1º passo é encontrar uma solução básica viável. O 2º passo consiste em calcular o valor das variáveis duais referentes a esta solução. O 3º passo, conhecido como *pricing*, é calcular o custo reduzido de cada variável e escolher uma variável associada a um custo reduzido negativo para entrar na base, caso isso não seja possível, encontramos a solução ótima. No 4º passo, encontramos um vetor que auxilia na escolha da variável que sairá da base. Com esse vetor calculado, no 5º passo devemos encontrar o maior número real que satisfaz uma determinada equação e esse valor real está associado a variável que sairá da base. O 6º e último passo consiste em atualizar a nova base e a nova solução básica assim como seu custo associado e voltamos para o 2º passo.

A decomposição de Dantzig-Wolfe, usada pela primeira vez na prática por Gilmore e Gomory (1961), transforma o problema em um outro equivalente com um número maior de variáveis, denominado *Problema Mestre*. Em seguida, aplica o Simplex revisado neste novo problema com uma diferença no passo 3 *(pricing)*, pois ao invés de efetuar o cáclculo do custo reduzido de cada variável, a decomposição de Dantzig-Wolfe escolhe a nova variável para entrar na base através de um problema de otimização associado ao problema

original.

O método da Geração de Colunas, como já relatado por Pizzolato e Gandolpho (2009), é uma extensão do Simplex revisado e da decomposição de Dantizig-Wolfe. O método da Geração de Colunas é indicado quando se tem um número muito grande de variáveis. Para iniciarmos o método resolvemos o problema original para um número relativamente pequeno de variáveis (chamado de *problema mestre restrito*) obtendo o valor de cada uma das variáveis e o valor das variáveis duais. Caso alguma variável possua valor nulo, esta não será usada na próxima iteração. Resolvemos o *subproblema de geração de colunas*, que é um subproblema dependente dos valores das variáveis duais encontrados que avalia os custos reduzidos de todas as variáveis indicando qual coluna possui custo reduzido mais negativo (considerando o problema original de minimização), ou seja, esta coluna está associada a uma variável que deverá ser incorporada ao problema na próxima iteração. O processo é repetido para o novo conjunto de variáveis e é parado quando não existir mais nenhuma coluna de custo reduzido negativo. Com intuito de aplicarmos o método da geração de colunas em um problema de programação linear ou inteira, diversos trabalhos usam a decomposição de Dantzig-Wolfe para o problema e aplicam o método da geração de colunas no problema mestre gerado. Dentre estes trabalhos podemos destacar: Senne et al. (2003) e Ribeiro e Lorena (2005).

Na prática, para diversos problemas, a aplicação do método da geração de colunas produz diversas colunas que não são utilizadas na solução final, o que pode gerar sérios problemas de convergência. Nesses casos, podemos observar que a as variáveis duais oscilam em torno da solução dual ótima. Estes problemas são chamados problemas de estabilização. Existem, na literatura, técnicas que previnem esse tipo de comportamento denominadas técnicas de estabilização. Em Briant et al. (2008), podemos encontrar descrições de algumas técnicas de estabilização.

Dentre as técinicas de estabiização mais conhecidas, destacamos: o método *Boxstep* (Marsten et al., 1975), o método *Analytic Centre Cutting Plane* (Du Merle et al., 1998), método de *Bundle* (Briant et al., 2008). Basicamente, o método *Boxstep* restringe o espaço de busca de soluções duais a uma região limitada onde a solução dual atual fica no centro. Já o método *Analytic Centre Cutting Plane* não permite mudanças muito drásticas entre duas soluções duais de uma iteratção para outra. Finalmente, o método de *Bundle* aproxima a região de busca das soluções duais a uma região delimitada por

uma função quadrática.

4.1 Decomposição de Dantzig-Wolfe

Consideremos o seguinte problema de programação linear denotado por $[PL]$:

$$\begin{aligned} Maximizar \quad & cx && (4.1)\\ Sujeito\ a \quad & Ax = b && (4.2)\\ & Dx = d && (4.3)\\ & x \geq 0 && (4.4)\\ & && (4.5) \end{aligned}$$

Qualquer problema de programação linear pode ser escrito dessa forma pois o uso de variáveis de folga transforma uma restrição de desigualdade em uma de igualdade.

Consideremos agora o conjunto R formado por todos os vetores $x \geq 0$ que satisfazem 4.3 e 4.4, ou seja,

$$R \;=\; \{x \mid x \geq 0 \ e \ Dx = d\} \tag{4.6}$$

É fácil ver que $[PL]$ é equivalente ao seguinte problema denotado por $[PL1]$

$$\begin{aligned} Maximizar \quad & cx && (4.7)\\ Sujeito\ a \quad & Ax = b && (4.8)\\ & x \in R && (4.9)\\ & && (4.10) \end{aligned}$$

Note que R é um poliedro. Suponhamos que este poliedro é limitado. Todo ponto contido em um poliedro limitado pode ser escrito como combinação convexa de seus pontos extremos. Por exemplo, na figura 4.1, a seguir, temos e um poliedro e podemos notar que o ponto A é combinação convexa dos pontos extremos $P2$, $P4$ e $P6$.

Assim, para todo x em R, temos:

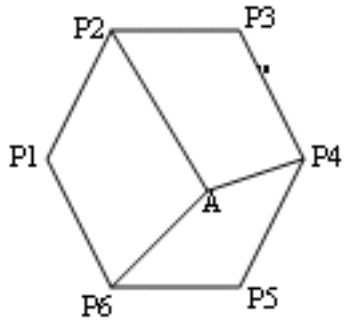

Figura 4.1: Poliedro limitado que mostra A como combinação convexa dos pontos extremos $P2$,$P4$ e $P6$

$$x = \sum_{j=1}^{Q} p_j \lambda_j \tag{4.11}$$

$$\sum_{j=1}^{Q} \lambda_j = 1 \tag{4.12}$$

$$\lambda \geq 0 \tag{4.13}$$

onde Q representa o número de pontos extremos e p_j representa o j-ésimo ponto extremo.

Assim, $[PL1]$ é equivalente ao seguinte problema, denotado por $[M]$

$$Maximizar \quad \sum_{j=1}^{Q} (cp_j)\lambda_j \tag{4.14}$$

$$Sujeito\ a \quad \sum_{j=1}^{Q} (Ap_j)\lambda_j = b \tag{4.15}$$

$$\sum_{j=1}^{Q} \lambda_j = 1 \tag{4.16}$$

$$\lambda \geq 0 \tag{4.17}$$

$$\tag{4.18}$$

$[M]$ é chamado de *problema mestre*. Em relação a $[PL1]$, $[M]$ possui menos restrições, e em geral, mais variáveis.

Sejam π e v os vetores que guardam os valores das variáveis duais associadas às restrições 4.16 e 4.17 do problema mestre $[M]$ respectivamente. O passo de *pricing* é resolvido de forma eficiente através do problema

$$Maximizar \quad (c - \pi A)x - v \tag{4.19}$$

$$Sujeito\ a \qquad Dx = d \tag{4.20}$$

$$x \geq 0 \tag{4.21}$$

$$\tag{4.22}$$

Se o custo ótimo do problema prévio é positivo então a solução ótima associada x^* é um ponto extremo do poliedro $[R]$ definido em 4.3 e a variável λ associada a x^* no problema mestre $[M]$ pode entrar para base. No caso do custo ótimo do problema ser nulo ou negativo ao invés de positivo, então a solução atual no problema mestre é ótima.

4.2 Geração de Colunas

Com intuito de ilustrar conceitos referentes a geração de colunas, propomos um simples problema. Tal problema é um caso particular do *Bin Packing* (Gilmore e Gomory, 1963) e o problema é descrito da seguinte maneira: Considere que uma tranportadora deverá fazer o transporte de 3 produtos diferentes: A, B e C. As caixas para transporte de produto tem capacidade padrão de 40 kg. O peso de cada produto está descrito na tabela 4.1

Tabela 4.1: Peso dos produtos A,B e C

Produto	Peso (Kg)
A	5
B	12
C	15

Seja U um número inteiro suficientemente grande de caixas tal que seja possível fazer o empacotamento de V, Y e Z produtos dos tipos A, B e C respectivamente que devem ser empacotados. O problema visa minimizar o número de caixas utilizadas. Com intuito de obtermos uma possível formulação, consideremos as seguintes variáveis:

$$t_j = \begin{cases} 1 & se\ a\ caixa\ j\ vai\ ser\ usada \\ 0 & C.C. \end{cases}$$

$$v_{ij} = \begin{cases} 1 & \textit{se o item i do tipo A vai na caixa j} \\ 0 & \textit{C.C.} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \textit{se o item i do tipo B vai na caixa j} \\ 0 & \textit{C.C.} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \textit{se o item i do tipo C vai na caixa j} \\ 0 & \textit{C.C.} \end{cases}$$

A formulação para essas variáveis é

$$\textit{Minimizar} \quad \sum_{j=1}^{U} t_j \tag{4.23}$$

$$\textit{Sujeito a} \quad \sum_{j=1}^{U} v_{ij} = 1 \ \forall i \in \{1, ..., V\} \tag{4.24}$$

$$\sum_{j=1}^{U} y_{ij} = 1 \ \forall i \in \{1, ..., Y\} \tag{4.25}$$

$$\sum_{j=1}^{U} z_{ij} = 1 \ \forall i \in \{1, ..., Z\} \tag{4.26}$$

$$\sum_{i=1}^{V} 5v_{ij} + \sum_{i=1}^{Y} 12y_{ij} + \sum_{i=1}^{Z} 15z_{ij} \leq 40t_j \ \forall j \in \{1, ..., U\} \tag{4.27}$$

$$\tag{4.28}$$

Os grupos de restrições 4.24, 4.25 e 4.26 são referentes ao fato de que todo produto deve ser empacotado por exatamente uma caixa, enquanto o grupo de restrições 4.27 é referente a respeitar a capacidade de cada caixa.

Esta formulação é baseada em Gilmore e Gomory (1963) e provou-se na prática ser não muito eficiente.

Uma outra formulação seria pensarmos em maneiras eficientes de se alocar os produtos na caixa. Por exemplo, uma maneira eficiente seria colocar 8 produtos do tipo A e nenhum dos outros tipos e esta maneira faz com que a caixa carregue sua capacidade máxima. Óbviamente existem diversas combinações para essas alocações. Na tabela 4.2 mostramos algumas possíveis maneiras.

Para cada maneira, criamos uma variável representando o número de caixas alocadas desta forma. Assim, por exemplo, a variável x_3 representa o número de caixas que leva 5 produtos do tipo A, 1 do tipo B e gera uma perda de 3 Kg por caixa.

Suponhamos que essa transportadora necessita transportar: 5000 produtos do tipo

Tabela 4.2: Algumas possíveis combinações para alocação dos produtos em uma caixa.

Tipo de produto	x_0	x_1	x_2	x_3	x_4	x_5
A	0	8	0	5	0	2
B	0	0	3	1	2	0
C	0	0	0	0	1	2
Perda	40	0	4	3	1	0

A, 3000 do tipo B e 2000 do tipo C. O objetivo do problema é minimizar o número de caixas necessária para esse transporte. O modelo completo é:

$$\begin{aligned}
Minimizar \quad & Z = x_0 + x_1 + x_2 + x_3 + x_4 + x_5 \\
Sujeitoa \quad & 0x_0 + 8x_1 + 0x_2 + 5x_3 + 0x_4 + 2x_5 \geq 5000 \\
& 0x_0 + 0x_1 + 3x_2 + 1x_3 + 2x_4 + 0x_5 \geq 3000 \\
& 0x_0 + 0x_1 + 0x_2 + 0x_3 + 1x_4 + 2x_5 \geq 2000 \\
& x_i \in \mathbb{Z}_+ \qquad \forall i \in \{0, ..., 5\}
\end{aligned}$$

A solução ótima para o problema é $x_0 = x_3 = x_2 = 0$, $x_1 = 563$, $x_4 = 1500$ e $x_5 = 250$ gerando um custo ótimo $Z^* = 2313$.

O nosso objetivo é resolver o problema via geração de colunas. Para isso, devemos inicialmente resolver o problema para um número reduzido de variáveis. Assim, escolhemos inicialmente x_0, x_1 e x_4. O mestre restrito é

$$\begin{aligned}
Minimizar \quad & Z = x_0 + x_1 + x_4 \\
Sujeito\ a \quad & 0x_0 + 8x_1 + 0x_4 \geq 5000 \\
& 0x_0 + 0x_1 + 2x_4 \geq 3000 \\
& 0x_0 + 0x_1 + 1x_4 \geq 2000 \\
& x_i \in \mathbb{Z}_+ \qquad \forall i \in \{0, 1, 4\}
\end{aligned}$$

A solução do mestre restrito prévio é $x_0 = 0$, $x_1 = 625$ e $x_4 = 2000$, enquanto o

valor das variáveis duais é $\pi_1 = \pi_2 = 0$ e $\pi_3 = 1$. O custo dessa solução $Z = 2625$, que é um limite superior para o custo ótimo $Z^* = 2313$.

Devemos agora resolver o subproblema de geração de colunas

$$Minimizar \quad W = 1 - \pi_1 v - \pi_2 y - \pi_3 z = 1 - z \tag{4.29}$$

$$Sujeito\ a \qquad 5v + 12y + 15z \leq 40 \tag{4.30}$$

$$v, y, z, p \in \mathbb{Z}_+ \tag{4.31}$$

$$\tag{4.32}$$

Onde v, y e z representam respectivamente o número de produtos do tipo A, B e C colocados em uma caixa.

O custo ótimo para esse problema é $W = -1$ e a solução ótima é $z = v = 2$ e $y = 0$ que corresponde a x_5. Assim, x_5 é incorporada ao modelo e x_0 é retirada do modelo pois obteve valor nulo.

O processo é repetido para o novo conjunto de variáveis e será parado quando o valor de W for nulo (solução ótima) ou positivo.

Capítulo 5

Algoritmos de busca direcionais

5.1 Algoritmos de descida

Toda teoria contida nesta seção foi baseada em Friedlander e Martínez (1992), Fritzsche (1977) e Friedlander (1994).

No contexto de otimização não-linear sem restrições, os métodos são baseados em direções de descida (para minimização) ou subida (para maximização).

Consideremos o seguinte problema

$$Minimizar \quad f(x), \quad x \in \mathbb{R}^n$$

Seja x^* uma solução ótima do problema em questão. Consideremos também x^k como uma estimativa de x^*, tal que $\nabla f(x^k) \neq 0$. Os algoritmos de busca direcional utilizam x^k e uma direção de subida ou descida para calcular o próximo iterado x^{k+1}:

Algoritmo 2 Algoritmo de busca direcional para minimização

Passo 1: Escolher $d_k \in \mathbb{R}^n$ tal que $\nabla^\intercal f(x^k) d_k < 0$.

Passo 2: Calcular $\lambda_k > 0$ *tal que* $f(x^k + \lambda_k d_k) < f(x^k)$ $(Busca\ linear)$.

Passo 3: Fazer $x^{k+1} \leftarrow x^k + \lambda_k d_k . x^k \leftarrow x^{k+1}$ até que $\nabla f(x^k)$ 0

O passo 1 consiste em escolher a direção de descida d_k. Notemos que essa direção não deve ser ortogonal ao gradiente pois caso contrário, não teríamos variação na função. O passo 2 é conhecido como busca linear e consiste em encontrar o tamanho do passo que garanta o decrescimento da função, o passo 3 consiste apenas em encontrar o novo elemento da sequência na direção escolhida.

O processo é parado quando para algum k, digamos k_0 temos $\nabla f(x^{k_0}) = 0$. Nesse caso ainda não se pode deduzir que x^{k_0} é solução do problema. Podemos dizer apenas que este é candidato a solução.

Consideremos por exemplo a função $f(x) = x^3$. Sabemos que para essa função $\nabla f(0) = 0$, porém 0 não é ponto de mínimo, nem de máximo dessa função (basta perceber que $f(-1) = -1 < f(0) = 0 < f(1) = 1$).

Consideremos agora um exemplo mostrado por Friedlander (1994): dada $f(x) = x^2$. Sabemos que $x^* = 0$ é o único minimizador dessa função, já que a função é convexa. A sequência definida por

$$x^k \quad = \quad 1 + \frac{1}{k}, com\, k \geq 1$$

pode ser gerada pelo algoritmo pois

$$f(x^{k+1}) = (1 + \frac{1}{k+1})^2 \quad \leq \quad (1 + \frac{1}{k})^2 = f(x^k).$$

No entanto,

$$\lim_{k\to\infty} x^k \quad = \quad 1 \tag{5.1}$$

Isto mostra que o fato da imagem de um elemento da sequência sempre ser menor que a imagem do seu antecessor, não necessariamente implica que a sequência convergirá para o mínimo da função. Isto acontece porque a distância entre x^{k+1} e x^k se aproxima de zero muito rapidamente.

Consideremos novamente a função $f(x) = x^2$. Suponhamos que ao aplicar o algoritmo 2 para determinado indice k_0 temos $x^{k_0} = -4$ e $x^{k_0+1} < 4$, onde x^{k_0+1} arbitrariamente próximo de 4. Nesse caso teríamos $f(x^{k_0}) > f(x^{k_0+1})$ porém o decréscimo é arbitrariamente pequeno. Caso os demais termos desta sequência sejam escolhidos de maneira semelhante o algoritmo terá sérios problemas de convergência pois temos passos grandes porém com pouco decréscimo na função. A figura 5.1 ilustra esse exemplo.

Uma terceira situação que pode atrapalhar a convergência do algoritmo 2 é se escolhemos direções d_k quase ortogonais ao gradiente $\nabla f(x^k)$ pois é sabido que direções ortognais ao gradiente não variam o valor da função.

Em resumo, existem três situações que atrapalham a convergência do algoritmo 2:

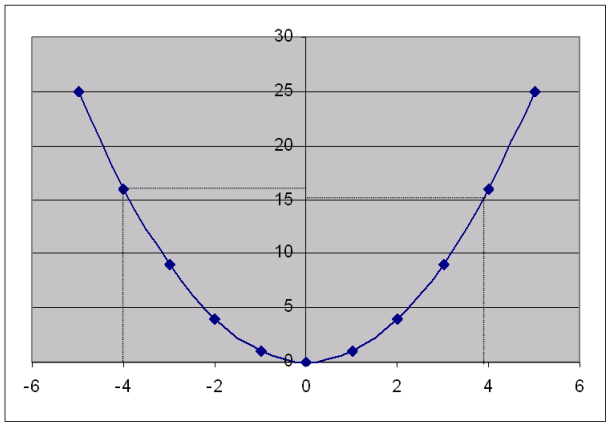

Figura 5.1: Gráfico exemplo para um possível problema de convergência

1. Distância entre x^{k+1} e x^k se aproximando de zero muito rapidamente.

2. Passos grandes com pouco decréscimo da função.

3. Direções d_k quase ortogonais ao gradiente $\nabla f(x^k)$.

Para cada uma destas, existe uma condição que é incorporada ao algoritmo. Estas condições são:

1. $\|d_k\| \geq \sigma \left\|\nabla f(x^k)\right\|, \forall k \in \mathbb{N}$, onde σ é uma constante positiva.

2. $f(x^k + \lambda_k d_k) < f(x^k) + \alpha \nabla^\intercal f(x^k) \lambda_k d_k$, $\forall k \in \mathbb{N}$,$\alpha \in (0,1)$ é uma constante.

3. $\nabla f(x^k) d_k \leq -\theta \left\|\nabla f(x^k)\right\|, \forall k \in \mathbb{N}$,onde $\theta \in (0,1)$ é uma constante.

Agora, definimos um algoritmo que incorpora todas as condições citadas. Se $x^k \in \mathbb{R}^n$ é tal que $\nabla f(x^k) \neq 0$, os passos para determinar x^{k+1} estão no algoritmo 3.

Algoritmo 3 Algoritmo com buscas direcionais com as condições incorporadas

Entrada: $\sigma > 0$, $\alpha \in (0,1)$ e $\theta \in (0,1)$.

Passo 1: Escolher $d_k \in \mathbb{R}^n$ tal que $\|d_k\| \geq \sigma \left\|\nabla f(x^k)\right\|$ e $\nabla f(x^k) d_k \leq -\theta \left\|\nabla f(x^k)\right\|$.

Passo 2: $\lambda \leftarrow 1$

Passo 3: Se $f(x^k + \lambda_k d_k) < f(x^k) + \alpha \nabla^\intercal f(x^k) \lambda_k d_k$ ir para o passo 5.

Passo 4: Escolher $\tilde{\lambda} \in [0,1\lambda \quad , \quad 0,9\lambda]$. Fazer $\lambda = \tilde{\lambda}$ e ir ao passo 3.

Passo 5: $\lambda_k \leftarrow \lambda$ e $x^{k+1} \leftarrow x^k + \lambda d_k$.

Do passo 2 em diante temos o que chamamos de *busca linear*.

O próximo teorema mostra que se a sequência x^k gerada pelo teorema 3 é tal que existe $\lim_{k\to\infty} x^k$, então este limite é um ponto estacionário da função (ponto cujo gradiente é nulo).

Teorema 5 *O algoritmo 3 pára com algum valor de k tal que* $\nabla f(x^k) = 0$*, ou gera uma sequência infinita* x^k *tal que qualquer ponto de acumulação dela é um ponto estacionário de* f*.*

Prova: Trata-se de um caso particular de um teorema mostrado em Friedlander e Martínez (1992) ■

Os dois métodos Quase-Newton mais conhecidos são: o DFP, proposto por Davidon, Fletcher e Powell, e mais usado até hoje: é o BFGS, proposto por Broyden, Fletcher, Goldfarb e Shanno.

5.2 Método do gradiente

O método do gradiente, como o próprio nome sugere, determina a direção de subida ou descida de acordo com o grtadiente da função objetivo: a) Quando o problema é de maximização escolhemos $d_k = \nabla f(x^k)$ pois $-\nabla f(x^k)$ aponta para o sentido de crescimento máximo da função. b) Quando o problema é de minimização escolhemos $d_k = -\nabla f(x^k)$ pois $-\nabla f(x^k)$ aponta para o sentido de decrescimento máximo da função. Tal escolha satisfaz trivialmente as duas propriedades do passo 1 do algoritmo 3. Assim, dado $x^k \in \mathbb{R}^n$ tal que $\nabla f(x^k) \neq 0$, os passos para encontrar x^{k+1} estão no algoritmo 4:

Algoritmo 4 Algoritmo com buscas direcionais baseado em um método do gradiente

Passo 1: $d_k \leftarrow -\nabla f(x^k)$.

Passo 2: Determinar λ_k minimizador de $f(x^k + \lambda d_k)$ sujeito a $\lambda \geq 0$.

Passo 3:$x^{k+1} \leftarrow x^k + \lambda d_k$.

O passo 2 é chamado de *Busca linear exata*.

Para verificar se o algoritmo 4 está bem-definido é necessário ter a certeza de que o subproblema do passo 2 tem sempre solução. Quando a função objetivo f é quadrática ou seja,

$$f(x) = \frac{1}{2}x^\intercal Gx + b^\intercal x + c. \tag{5.2}$$

Onde G é uma matriz definida positiva, b é um vetor constante e c uma constante, a busca linear exata fornece

$$\lambda_k = \frac{\nabla^\intercal f(x^k)\nabla f(x^k)}{\nabla^\intercal f(x^k)G\nabla f(x^k)}. \tag{5.3}$$

O próximo teorema aborda a convergência do método do gradiente para o caso de funções objetivo quadráticas, mostrando que o minimizador global é encontrado quando número de iterações tende ao infinito.

Teorema 6 *Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função quadrática. Seja x^* o minimizador global de f. Dado $x^0 \in \mathbb{R}^n$ arbitrário. A sequência x^k gerada pelo algoritmo 4 é tal que:*

1. $\lim_{k\to\infty} x^k = x^*$
2. $\lim_{k\to\infty} f(x^k) = f(x^*)$

Prova: Ver Luenberger e Ye (2008) ■

5.3 Método de Newton

O algoritmo para o método de Newton se diferencia do algoritmo 4 somente no passo 1 pois no método de Newton devemos calcular d_k de modo que

$$\nabla^2 f(x^k) \quad d_k = -\nabla f(x^k), \tag{5.4}$$

onde $\nabla^2 f(x^k)$ é a matriz hessiana da função f avaliada no ponto x^k. Óbviamente, este passo pode não estar bem-definido caso $\nabla^2 f(x^k)$ seja singular pois o sistema prévio, nesse caso, não teria solução. A idéia consiste em aproximar f de forma quadrática em torno de cada ponto x^k.

Na literatura, existe um importante resultado com relação a convergência local do método de Newton. Tal resultado é mostrado no teorema 7.

Teorema 7 *Seja $f : \mathbb{R}^n \to \mathbb{R}, f \in C^3$, onde C^3 é o conjunto das funções deriváveis no mínimo até 3^a ordem. Seja x^* um minimizador local de f em $\mathbb{R}^n$, tal que $\nabla^2 f(x^*)$ é definida positiva. Então existe $\epsilon > 0$ tal que se $x^0 \in \mathbb{B}(x^*, \varepsilon)$, onde $\mathbb{B}(x^*, \varepsilon) = \{y \in \mathbb{R}^n | d(x^*, y) < \epsilon\}$, e o tamanho do passo na iteração k $\lambda_k = 1$ para todo $k \in \mathbb{N}$, A sequência x^k gerada quando aplicamos o método de Newton verifica:*

1. $\nabla^2 f(x^k)$ *é definida positiva para todo* $k \in \mathbb{N}$.

2. $\lim_{k\to\infty} x^k = x^*$

3. *Existe* $c \in (0,1)$ *tal que* $\left\|x^{k+1} - x^*\right\|^2 \leq c\left\|x^k - x^*\right\|^2$,*para todo* $k \in \mathbb{N}$

Prova: Ver Luenberger e Ye (2008) ■

Cada item do teorema 7 mostra que se escolhemos x^0 próximo de x^* podemos afirmar :

1. O passo relativo a escolha do d_k no método de Newton está bem definido.

2. A sequência converge para o minimizador local da função.

3. A ordem de convergência é no mínimo quadrática.

Quando o método de Newton é aplicado e a escolha da direção d_k é feita de modo que $\nabla^2 f(x^k) \quad d_k = -\nabla f(x^k)$ satisfaz às duas condições do passo 1 do algoritmo 3 e também às condições iniciais do Teorema 7, os elementos da sequência x^k chegam tão próximos de x^* quanto queremos.

É válido ressaltar que para o método de Newton, se faz necessária a resolução do sistema $\nabla^2 f(x^k)d_k = -\nabla f(x^k)$ o que pode ter um custo alto em termos computacionais. Por isso, são mais usados os métodos quase-newton onde a hessiana é aproximada.

5.4 Métodos Quase-Newton

Os métodos Quase-Newton se caracterizam pela escolha de direção de descida d_k de modo que

$$d_k \quad = \quad -H_k \nabla f(x^k). \tag{5.5}$$

Onde $H_k \in \mathbb{R}^{n\times n}$ é uma matriz simétrica que aproxima a hessiana no ponto x^k sem o "overhead"de ter que inverter a hessiana.

Alguns exemplos de aproximação:

a) diagonalização escalada:

$$d_k = -D^{(k)}\nabla f(x^k),$$

onde $[D^{(k)}]_i = (\frac{\partial^2 f(x^k)}{\partial (x_i)^2})^{-1}$

b) Newton modificado:

$$d_k = -D^{(k)}\nabla f(x^k),$$

onde $D^{(k)} = (\nabla^2 f(x^{(0)}))-1$, determinado geralmente por diferenças finitas.

c) Newton discretizado:

$$d_k = -D^{(k)}\nabla f(x^k),$$

onde $D^{(k)} = (H(x^{(k)}))-1$. $H(x)$ representa uma aproximação simétrica e positiva para a hessiana.

O algoritmo 5 descreve os passos do método de Quase-Newton.

Algoritmo 5 Método Quase-Newton

Entrada: H_0 matriz simétrica e λ_0.

Passo 1: $d_k \leftarrow -H_k \nabla f(x^k)$.

Passo 2: Determinar λ_k através de uma busca linear.

Passo 3:$x^{k+1} \leftarrow x^k + \lambda d_k$.

Passo 4: Determinar H_{k+1} simétrica de modo que

$$H_{k+1}(\nabla f(x^{j+1}) - \nabla f(x^j)) = x^{j+1} - x^j, \textit{ para todo } j \leq k.$$

Quando a função objetivo é quadrática, apesar de não estarmos citando a prova nesse trabalho, o método Quase-Newton encontra x^* em um número finito de iterações (ver Luenberger e Ye (2008)) e sem precisar ter informações completas sobre a matriz hessiana, sendo ainda $H_n = G^{-1}$, onde $G = \nabla^2 f(x^n)$. Portanto $d_n = -G^{-1}(\nabla f(x^n))$ e $x^{n+1} = x^*$.

Existem diversos métodos Quase-Newton e a diferença entre eles é como é feito o ajuste da matriz H_{k+1} em função da matriz H_k na iteração k. Os dois métodos Quase-Newton mais conhecidos são: o primeiro e mais usado até hoje foi proposto por Davidon,

Fletcher e Powell e o asegundo foi porposto por Broyden, Fletcher, Goldfarb e Shanno, que são denotados por (DFP) e $(BFGS)$ respecitvamente (as iniciais dos autores).

No método (DFP),

$$H_{k+1} = H_k + \frac{p_k p_k^\intercal}{p_k^\intercal q_k} - \frac{H_k q_k q_k^\intercal H_k}{q_k^\intercal H_k q_k},$$

enquanto para o método $(BFGS)$, temos

$$H_{k+1} = H_k + (\frac{1 + q_k^\intercal H_k q_k}{q_k^\intercal p_k})\frac{p_k p_k^\intercal}{p_k^\intercal p_k} - \frac{p_k q_k^\intercal H_k + H_k q_k p_k^\intercal}{q_k^\intercal p_k}, \tag{5.6}$$

onde $p_k = \lambda_k d_k = x^{k+1} - x^k$ *e* $q_k = \nabla f(x^{k+1}) - \nabla f(x^k)$.

O método Quase-Newton mais utilizado é o método $(BFGS)$. Basicamente, no método $(BFGS)$, em cada iteração, H_{k+1} consiste numa correção de posto 1 da matriz H_k e, em particular, B_{k+1} é a projeção ortogonal de B_k no conjunto das matrizes que satisfazem a equação secante, considerando a norma de Frobeinus ((DA et al.)). Cada método Quase-Newton utiliza uma maneira distinta de contribuir a informação sobre a curvatura hessiana $[(\nabla^2 f(x^{j+1})(x^{j+1} - x^j)]$ dentro das matrizes H^k.

Capítulo 6

O Problema da Alocação Generalizada

6.1 Definição e Aplicações

O PAG é definido matematiamente da seguinte maneira: Sejam $I = \{1, .., m\}$ um conjunto de máquinas e $J = \{1, .., n\}$ um conjunto de tarefas. Para cada maquina em $i \in I$, é dado o tempo disponível b_i. Para cada máquina $i \in I$ e cada tarefa $j \in J$, são dados o custo c_{ij} de alocar a tarefa j na máquina i e o tempo a_{ij} requerido para processar a tarefa j na máquina i. O objetivo do problema é atribuir cada tarefa a exatamente uma máquina de modo que o total de tempo alocado em uma máquina não ultrapasse o tempo disponível dela e minimizando o custo total de atribuição.

Abaixo, nós mostramos uma formulação de Programação inteira básica para o PAG que usa variáveis binárias x_{ij}, indicando que a tarefa j está atribuida à máquina i, para cada $i \in I$ e $j \in J$.

$$\min \quad \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} \tag{6.1}$$

$$\text{sujeito a} \quad \sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i \quad i \in I \tag{6.2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad j \in J \tag{6.3}$$

$$x_{ij} \in \{0, 1\} \quad i \in I,\, j \in J. \tag{6.4}$$

Para facilitar a compreensão do PAG e da nossa técnica de suavição da função

dual lagrangeana, alguns passos e notações desta serão mostrados com um auxílio de uma pequena instância. A instância tem 2 máquinas ($m = 2$) e 4 tarefas ($n = 4$). Para todo $i \in \{1, 2\}$ e $j \in \{1, ..., 4\}$, o custo c_{ij}, o tempo a_{ij} e os tempos disponíveis b_1 e b_2 nas máquinas são mostrados abaixo:

$$C = [c_{ij}] = \begin{bmatrix} 10 & 15 & 25 & 4 \\ 11 & 52 & 8 & 16 \end{bmatrix} \quad A = [a_{ij}] = \begin{bmatrix} 9 & 2 & 7 & 5 \\ 8 & 3 & 4 & 7 \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 12 \end{bmatrix}$$

Uma possível solução para essa instância seria alocar as tarefas 2 e 4 na máquina 1 e as tarefas 1 e 3 na máquina 2. A figura 6.1 mostra ao gráfico de Gantt para esta solução.

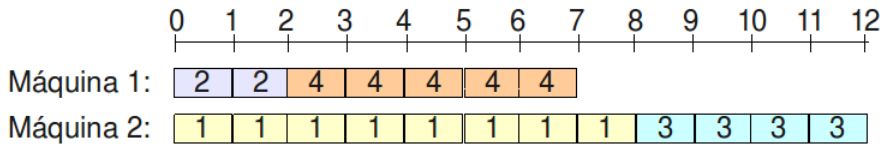

Figura 6.1: Gráfico de Gantt para a possível solução da instância do PAG.

Notemos que essa solução é viável pois cada tarefa foi alocada a exatamente uma máquina e a capacidade de cada uma das máquinas não foi violada. O custo dessa solução é $c_{12} + c_{14} + c_{21} + c_{23} = 15 + 4 + 11 + 8 = 38$.

6.2 Relaxação Lagrangeana para o PAG

Na Relaxação Lagrangeana usada neste trabalho (e também usada por Fisher et al. (1986); Guignard e Rosenwein (1989); Pigatti et al. (2005)), nós dualizamos as restrições (6.3) usando os multiplicadores π_j, para todo $j \in J$. Isto nos leva ao seguinte subproblema lagrangeano:

$$[L(\pi)]$$

$$\min \quad \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} + \sum_{j=1}^{n}\pi_j(\sum_{i=1}^{m} x_{ij} - 1) \tag{6.5}$$

$$\text{sujeito a} \quad \sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i \qquad i \in I \tag{6.6}$$

$$x_{ij} \in \{0, 1\} \qquad i \in I,\ j \in J. \tag{6.7}$$

Reescrevendo (6.5), nós obtemos $[L(\pi)]$ equivalente a:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+\pi_j)x_{ij} - \sum_{j=1}^{n}\pi_j \\ sujeito\,a \quad & (6.2)\,e\,(6.4). \end{aligned}$$

O correspondente problema dual lagrangeano é:

$$[LD] \max_{\pi} L(\pi), \tag{6.8}$$

Onde $L(\pi)$ denota o valor da solução ótima do subproblema $[L(\pi)]$.

Seguindo a decomposição lagrangeana aproximada por Guignard e Kim (1987), nós podemos decompor $[L(\pi)]$ em m subproblemas. Para cada $i \in I$, nós resolvemos:

$$\begin{aligned} & [KP(\pi)^i] \\ & \min \quad \sum_{j=1}^{n}(c_{ij}+\pi_j)x_{ij} \\ & \text{sujeito a} \quad (6.2)\,e\,(6.4). \end{aligned}$$

Cada subproblema acima é um problema da mochila booleano (KP) (Kellerer et al., 2004) e pode ser resolvido por programação dinâmica em tempo $O(b_i n)$, com a usual recursão baseada em (3.4):

$$OPT_i(l,k) = \begin{cases} \min\{OPT_i(l-1,k), OPT_i(l-1,k-a_{il}) + c_{il} + \pi_l\}, & \text{se } k \geq a_{il} \text{ e } l > 1, \\ OPT_i(l-1,k), & \text{se } k < a_{il} \text{ e } l > 1, \\ \min\{0, c_{il}+\pi_l\}, & \text{se } k \geq a_{il} \text{ e } l = 1, \\ 0, & \text{se } k < a_{il} \text{ e } l = 1, \end{cases} \tag{6.9}$$

para todo $l \in J$ e $k \in \{0, ..., b_i\}$, onde $OPT_i(l,k)$ denota o custo de um problema da mochila considerando somente os itens $1, ..., l$ e capacidade k.

Como resultado, nós temos $KP(\pi)^i = OPT_i(n, b_i), \forall\, i \in I$. Então, resolver o subproblema lagrangeano $[L(\pi)]$ se reduz a resolver m KPs, resultando o seguinte limite inferior para o problema original: $L(\pi) = \sum_{i=1}^{m} KP(\pi)^i - \sum_{j=1}^{n}\pi_j$, ou equivalentemente, $L(\pi) = \sum_{i=1}^{m} OPT_i(n, b_i) - \sum_{j=1}^{n}\pi_j$.

Capítulo 7

A função dual lagrangeana suave

Como já mencionamos na introdução desta dissertação, quando submetemos problemas de otimização combinatória ao método da Relaxação Lagrangeana a função dual lagrangeana não é diferenciável. No capítulo 2 apresentamos um simples problema de otimização combinatória, onde a função dual deste problema é $L(\theta) = min\{2 + 15\theta, 10 - 5\theta\}$, que não é diferenciável devido a função $f(x, y) = min\{x, y\}$.

Neste capítulo, nós formalizamos uma aproximação suave para $f(x, y) = min\{x, y\}$ com intuito de obter uma aproximação suave para $L(\theta)$ ou $L(\pi)$.

Nós denotamos a aproximação de f por $\tilde{f}$ que é definida da seguinte maneira:

$$\tilde{f}(x, y) = \frac{x + y - \sqrt{(x - y)^2 + 4\delta^2}}{2}, \tag{7.1}$$

onde δ é o erro de aproximação.

Alguns trabalhos que propõem suavizações semelhantes podem ser encontrados na literatura. Por exemplo, Di Bello (2005) usou uma função suave muito semelhante a $\tilde{f}$ em um modelo de análise de comportamento da umidade do solo, onde a função

$$\psi(x_t, M, d) = \frac{x_t - M + \sqrt{(x_t - M)^2 + 4d^2}}{2}, \tag{7.2}$$

foi usada como suavização para a função

$$R_t(M, x_t) = \begin{cases} 0, & \text{se } x_t \leq M, \\ x_t - M & \text{se } x_t > M, \end{cases} \tag{7.3}$$

onde M é uma constante do método e d é o erro de aproximãção da suavização.

Moreno et al. (2008) usou duas funções suaves em um problema de classificação: a 1ª delas foi

$$\alpha(y,\beta) \;=\; \frac{y+\sqrt{y^2+\beta^2}}{2}, \tag{7.4}$$

usada para suavizar com erro positivo β a função

$$\omega(y) \;=\; max\{y,0\}, \tag{7.5}$$

e a 2ª suavização foi

$$\mu(x,y,\beta) \;=\; \sqrt{\|x-y\|_2^2+\beta^2}, \tag{7.6}$$

usada para suavizar com erro positivo β a função

$$\gamma(x,y) \;=\; \|x-y\|_2\,. \tag{7.7}$$

Agora, como exemplo de ilustração, calculamos $min\{-2,4\}$ aproximadamente utilizando a função $\tilde{f}$ para diversos erros de aproximação δ. Os resultados foram colocados na tabela 7.1.

Como já era esperado, observando a tabela 7.1 notamos que quanto menor o erro δ, o valor de $\tilde{f}(-2,4)$ se aproxima do valor de $min\{2,4\}$.

Se aproximamos a função dual lagrangeana do nosso exemplo $L(\theta) = min\{2+15\theta, 10-5\theta\}$ inferiormente por uma função suave controlada por um erro δ, temos uma aproximação suave para a função dual lagrangeana. A figura 7.1 representa a função dual lagrangena $L(\theta) = min\{2+15\theta, 10-5\theta\}$ e três aproximações com erros $\delta = 1$, $\delta = 2$ e $\delta = 4$, que são denotadas L, L1, L2, L4 respectivamente.

Notemos que a medida que o erro δ decresce o máximo da função aproximada se aproxima do máximo da função dual $L(\theta)$.

Para qualquer $\delta > 0$, a função aproximação $\tilde{f}$ tem as seguintes propriedades:

- $\tilde{f}$ é concava.
- $f(x,y) - \delta < \tilde{f}(x,y) < f(x,y)$.

Tabela 7.1: Cálculo de $min(-2, 4)$ utilizando $\tilde{f}$ para diversos erros δ

δ	$\tilde{f}(-2, 4)$
5	-4,830951895
3,5	-3,609772229
2,45	-2,873306081
1,715	-2,455607761
1,2005	-2,231284613
0,84035	-2,115475585
0,588245	-2,057128093
0,4117715	-2,028127436
0,28824005	-2,013815244
0,201768035	-2,006777401
0,141237625	-2,003322838
0,098866337	-2,00162865
0,069206436	-2,000798149
0,048444505	-2,00039112
0,033911154	-2,000191655
0,023737808	-2,000093912
0,016616465	-2,000046017
0,011631526	-2,000022549
0,008142068	-2,000011049
0,005699448	-2,000005414
0,003989613	-2,000002653

- $\lim_{(x-y)\to+\infty} \tilde{f}(x, y) = y.$
- $\lim_{(x-y)\to-\infty} \tilde{f}(x, y) = x.$
- $\frac{\partial \tilde{f}(x,y)}{\partial x} = \frac{1}{2} - \frac{2(x-y)}{4\sqrt{(x-y)^2+4\delta^2}} = \frac{1}{2}\left(1 - \frac{x-y}{\sqrt{(x-y)^2+4\delta^2}}\right).$
- $x = y \Rightarrow \frac{\partial \tilde{f}(x,y)}{\partial x} = \frac{1}{2}.$
- $\frac{\partial \tilde{f}(x,y)}{\partial x} + \frac{\partial \tilde{f}(x,y)}{\partial y} = 1 - \left(\frac{x-y+y-x}{\sqrt{(x-y)^2+4\delta^2}}\right) = 1.$

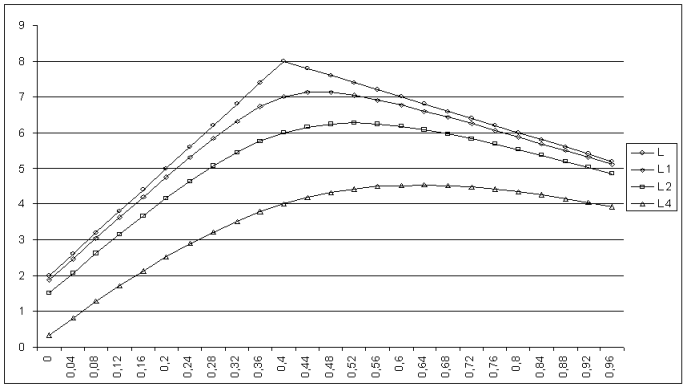

Figura 7.1: Função dual lagrangeana $L(\theta) = min\{2 + 15\theta, 10 - 5\theta\}$ e as correspondentes funções aproximações suaves para $\delta = 1, 2$, e 4.

As provas para as propriedades prévias estão no Apêndice A.

Observando a recursão (7.3), nós podemos notar que a função dual lagrangeana $L(\pi)$ também não é diferenciável somente devido a função $f(x, y) = min\{x, y\}$.

Suavizando $L(\pi)$ substituindo $f(x, y) = min\{x, y\}$ por $\tilde{f}$, nós obtemos a seguinte aproximação para a recursão (7.3):

$$\widetilde{OPT_i}(l,k) = \begin{cases} \tilde{f}(\widetilde{OPT_i}(l-1,k), \widetilde{OPT_i}(l-1,k-a_{il}) + c_{il} + \pi_l), & \text{se } k \geq a_{il} \text{ e } l > 1, \\ \widetilde{OPT_i}(l-1,k), & \text{se } k < a_{il} \text{ e } l > 1, \\ \tilde{f}(0, c_{il} + \pi_l), & \text{se } k \geq a_{il} \text{ e } l = 1, \\ 0, & \text{if } k < a_{il} \text{ and } l = 1, \end{cases} \tag{7.8}$$

Para todo $l \in J$ e $k \in \{0, ..., b_i\}$, onde $\widetilde{OPT_i}(l, k)$ denota o custo aproximado de um problema da mochila considerando somente os itens $1, ..., l$ e capacidade k.

Assim, o limite final (6.2) é aproximado por:

$$\widetilde{L}(\pi) = \sum_{i=1}^{m} \widetilde{OPT_i}(n, b_i) - \sum_{j=1}^{n} \pi_j.$$

Para a instância apresentada com $\delta = 1,75$ e

$$\pi = \begin{bmatrix} -30,6568 \\ -31,7143 \\ -30,7229 \\ -28,4097 \end{bmatrix}$$

nós temos

$$\widetilde{OPT_1} = \begin{bmatrix} 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & -112,6 & -112,6 \\ 0,0 & 0,0 & -111,9 & -111,9 & -111,9 & -111,9 & -111,9 & -111,9 & -111,9 & -114,0 & -114,0 \\ 0,0 & 0,0 & -111,9 & -111,9 & -111,9 & -111,9 & -111,9 & -112,1 & -112,1 & -209,8 & -209,8 \\ 0,0 & 0,0 & -111,9 & -111,9 & -111,9 & -112,8 & -112,8 & -221,5 & -221,5 & -221,7 & -221,7 \end{bmatrix}$$

Onde o valor de $\widetilde{OPT}_1(l,t)$ é localizado na l-ésima linha $(t+1)$-ésima coluna.

Capítulo 8

Computando o gradiente de $\widetilde{L}(\pi)$.

Agora, nós mostramos como calcular $\nabla\widetilde{L}(\pi)$ e com esse intuito apresentamos algumas importantes notações:

- $S^i_{q,k} \longrightarrow$ o conjunto de todos os subconjuntos Q de $J_q = \{q, ..., n\}$ cuja soma $\sum_{l \in Q} a_{il} = k$, onde $J_{n+1} = \emptyset$.
- $k_i(Q, l) = \sum_{p \in Q \cap \{l+1,...,n\}} a_{ip}$
- $\tilde{g}_x(x, y) = \frac{\partial \tilde{f}(x,y)}{\partial x}$.
- $\tilde{g}_y(x, y) = \frac{\partial \tilde{f}(x,y)}{\partial y}$.
- $h^{i,k}_{x,l} = \tilde{g_x}(\widetilde{OPT_i}(l-1, b_i - k), \widetilde{OPT_i}(l-1, b_i - k - a_{i,l}) + c_{il} + \pi_l)$, se $k \leq b_i - a_{il}$.
- $h^{i,k}_{x,l} = 1$, se $k > b_i - a_{il}$.
- $h^{i,k}_{y,l} = \tilde{g_y}(\widetilde{OPT_i}(l-1, b_i - k), \widetilde{OPT_i}(l-1, b_i - k - a_{i,l}) + c_{il} + \pi_l)$, se $k \leq b_i - a_{il}$.
- $h^{i,k}_{y,l} = 0$, se $k > b_i - a_{il}$.

Para a instância apresentada no capítulo 6, considerando $\pi = (-30,6568; -31,7143; -30,7229;$ e $\delta = 1,75$. Para ilustrar estas notações, nós apresentamos alguns resultados:

- $S^1_{2,7} = \{\{3\}, \{2, 4\}\}$
- $k_i(\{2, 4\}, 2) = 5$
- $h^{1,0}_{x,2} = \widetilde{g}_x(-112, 6; 0) = 0,612184$ (ver $\tilde{OPT}_1$ na seção 6.2)

Com o intuito de computar $\nabla\widetilde{L}(\pi)$, nós usamos uma matriz auxiliar que tem as mesmas dimensões da matriz $\widetilde{OPT_i}$. Esta matriz, denotada por AUX_i, também nos fornece uma solução fracionária associada à $\widetilde{OPT_i}(n, b_i)$. Cada célula $AUX_i(l, k)$ representa a fração da solução parcial $\widetilde{OPT_i}(l, k)$ usada na solução final. Tendo esta definição em mente é fácil ver que:

$$AUX_i(n,k) \;=\; \begin{cases} 1 & \text{se } k = b_i, \\ 0 & \text{se } k \neq b_i. \end{cases}$$

Em outras palavras, $\widetilde{OPT_i}(n, b_i)$ é usada inteiramente na solução final. Para todo $l \in \{0, .., n-1\}, k \in \{0, ..., b_i\}$, nós definimos:

$$AUX_i(l, b_i - k) \;=\; h^{i,k}_{x,l+1} \times AUX_i(l+1, b_i - k) + h^{i,k-a_{i(l+1)}}_{y,l+1} \times AUX_i(l+1, b_i - k + a_{i(l+1)})$$

onde $AUX_i(l, t)$ é definida como zero para todo $l \in \{1, .., n\}$ e $t > b_i$.

É fácil ver que a matriz AUX_i pode também ser preenchida em tempo $O(b_i n)$. A matriz AUX_1 para a instância apresentada no capítulo 6 (ver matriz $\tilde{OPT}_1$ na seção 6.2) segue abaixo:

$$AUX_1 = \begin{bmatrix} 0 & 0,0201373 & 0 & 0,979626 & 0 & 0,00023006 & 0 & 0 & 2,50 \times 10^{-6} & 0 & 3,95 \times 10^{-6} \\ 0 & 0 & 0 & 0,0201421 & 0 & 0,979851 & 0 & 0 & 0 & 0 & 6,46 \times 10^{-6} \\ 0 & 0 & 0 & 0 & 0 & 0,979851 & 0 & 0 & 0 & 0 & 0,0201485 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Onde o valor de $AUX_1(l, t)$ é localizado na l-ésima linha and $(t+1)$-ésima coluna.

Note que para $k = 7$, $i = 1$ e $l = 1$ nós temos:

$$\begin{aligned} AUX_1(1,3) \;&=\; h^{1,7}_{x,2} \times AUX_1(2,3) + h^{1,5}_{y,2} \times AUX_1(2,5) \\ &\cong\; 0,9796 \end{aligned}$$

Para todo $j \in J$, nós definimos a fração do item j que é colocada na mochila i, denotada por x_{ij}, como:

$$x_{ij} \;=\; \sum_{k=0}^{b_i} AUX_i(j, b_i - k) \times h^{i,k}_{y,j}. \tag{8.1}$$

A matriz x_{ij} da instância apresentada no capítulo 6 é:

$$x_{ij} = \begin{bmatrix} 3,95 \times 10^{-6} & 0,999761 & 0,0201421 & 0,979851 \\ 0,983557 & 0,00394995 & 0,998174 & 0,0140864 \end{bmatrix}$$

Como exemplo, nós mostramos como calcuar $x_{1,3}$:

$$\begin{aligned} x_{1,3} &= \sum_{k=0}^{10} AUX_1(3, 10-k) \times h_{y,3}^{i,k} \\ &\cong 0,0201421 \end{aligned}$$

O teorema abaixo é o mais importante dessa dissertação pois mostra que cada componente do vetor gradiente é igual a violação da restrição dualizada associada, que é uma propriedade análoga ao teorema 4 da teoria lagrangeana clássica (ver prova no capítulo 2). Portanto, se o gradiente é zero, nós temos uma solução fracionária viável.

Teorema 8 $\nabla \widetilde{L}(\pi) = v$, *onde* $v_j = \sum_{i=1}^{m} x_{ij} - 1, \forall j \in J$.

Prova: Ver apêndice B. ■

Capítulo 9

Resolvendo o problema dual

9.1 Algoritmo de Resolução

Para ajustar os multiplicadores de Lagrange de (6.8), nós escolhemos um método Quase-Newton proposto por Meza et al. (2007). A matriz simétrica que aproxima a hessiana é atualizada como no método Quase-Newton BFGS. O algortimo padrão método Quase-Newton é o algoritmo 5 e os cálculos necessários para atualizar a aproximação da matriz hessiana estão mostrados em 5.6. A idéia inicial era aplicação de um método gradiente de subida aproximado porém experimentos mostraram resultados mais satisfatórios quando aplicamos o método Quase-Newton.

É válido ressaltar que em um primeiro momento não poderíamos aplicar o método de Newton pois para a aplicação deste são necessários conhecimentos sobre a segunda derivada da função assim como conhecimentos sobre a matriz Hessiana desta.

A cada iteração do método Quase-Newton, os valores dos multiplicadores são atualizados, com o objetivo de maximizar a função dual lagrangeana e diminuir o valor da norma $||\nabla\widetilde{L}(\pi^k)||$.

O método Quase-Newton para quando $||\nabla\widetilde{L}(\pi^k)||$ ou a norma relativa do gradiente sejam menores do que um dado parâmetro ϵ_0.

Os parâmetros de entrada H_0 e λ_0 são respectivamente I_n e 1, onde I_n é a matriz identidade de ordem n, sendo n número de tarefas da instância corrente.

9.2 Existência de um ponto de gradiente zero

Para o PAG, pode acontecer que o conjunto de multiplicadores de Lagrange ótimos (para a função dual lagrangeana original) seja ilimitado. Neste caso, um ponto de gradiente zero pode não existir. Consequentemente, com intuito de provar que tal ponto existe, nós usamos as seguintes condições nas instâncias do PAG.

Condição 1 *Para todo $j \in J$, existe uma solução do subproblema onde cada tarefa é processada em exatamente uma máquina, exceto para a tarefa j, que é processada em duas máquinas.*

Condição 2 *Para todo $j \in J$, existe uma solução do subproblema onde cada tarefa é processada em exatamente uma máquina, exceto para a tarefa j, que não é processada em máquina alguma.*

Note que a condição 2 é sempre satisfeita por instâncias viáveis, mas a condição 1 pode ser falsa. Neste caso, o nosso método não funcionará. Apesar disso, nós verificamos que a condição 1 é satisfeita para todas as instâncias utilizadas em nossos experimentos retiradas do amplamente usado *OR-Library*. O seguinte teorema implica na existência de um ponto de gradiente zero para a função $\tilde{L}(\pi)$, dado que a mesma é côncava, contínua e diferenciável.

Teorema 9 *Para qualquer π^0, $S = \{\pi \in \mathbb{R}^n | \widetilde{L}(\pi) \geq \widetilde{L}(\pi^0)\}$ é limitado.*

Para cada solução da condição 1, consideremos a função $\widetilde{L}_j^+$ que representa o custo associado da solução para o multiplicador π: $\widetilde{L}_j^+(\pi) = p_j - \pi_j$, onde p_j é uma constante. Para cada solução da condição 2, consideremos a função $\widetilde{L}_j^-$ que representa o custo associado da solução para o multiplicador π: $\widetilde{L}_j^-(\pi) = q_j + \pi_j$, onde q_j é uma constante.

Note que, para qualquer $j \in J$ e qualquer $\pi \in \mathbb{R}^n$, nós temos $L(\pi) \leq \widetilde{L}_j^+(\pi)$ e $L(\pi) \leq \widetilde{L}_j^-(\pi)$. Nós usamos este fato para demonstrar que $\widetilde{L}_j^+$ e $\widetilde{L}_j^-$ limitam a trajetória do método. Para isso, Seja π^t o multiplicador na iteração t. Para qualquer $k \in J$, existe $\pi(k) \in \mathbb{R}^n$ tal que $\widetilde{L}_k^+(\pi(k)) = L(\pi^0)$. Assim, nós temos $\pi_k^t \leq \pi_k(k)$, caso contrário nós teríamos $L(\pi^t) < L(\pi^0)$, que não é possível desde que nosso método seja um método gradiente de subida.

Como consequência, a trajetória do nosso método tem um limite superior, para qualquer componente $k \in J$. Similarmente, usando $\widetilde{L}_k^-$ pode nos mostrar que π_k^+ tem um limite inferior, para qualquer $k \in J$, o que conclui nossa prova. ■

É interessante notar que, desde que a função dual seja côncava (veja teorema 10), a solução viável fracionária encontrada está associada ao valor ótimo da função dual lagrangeana suavizada.

Capítulo 10

Resultados computacionais

Nosso método foi testado em instâncias do PAG encontradas em OR-Library. Existem 5 tipos de instâncias no OR-Library:A, B, C, D e E, designadas como no seguinte exemplo: d05100 é uma instância do tipo D com 5 máquinas e 100 tarefas. Nós começamos o método com um erro δ relativamente grande, aplicamos o método Quase-Newton e ao seu término multiplicamos o valor de δ por um fator η, com $\eta \in (0, 1)$. Paramos esse procedimento quando $\delta < 0,001$. No próximo algoritmo, mostramos esses passos.

Algoritmo 6 Suavizando a função dual do PAG

Entrada: H_0 matriz simétrica, δ, η

enquanto $\delta \geq 0,001$ **faça**

$QUASENEWTON(H_0, \pi^k, \lambda_k, H_k, \epsilon_0)$

$LB \leftarrow \tilde{L}(\pi^k)$

$\delta \leftarrow \eta\delta$

fim enquanto

Onde a função $QUASENEWTON$ representa os passos do método Quase-Newton descritos no algoritmo 5, atualizando a matriz H_k conforme o método Quase-Newton ($BFGS$). O número de repetições dos passos acima é referenciado por iterações externas e o número de iterações gastas pelo método Quase-Newton é representado por iterações internas.

Para os nossos testes consideramos $\delta = 5$ como parâmetro inicial. Para o parâmetro η foram feitos testes considerando 3 valores diferentes: $\eta = 0,3$, $\eta = 0,5$ e $\eta = 0,7$.

Para o parâmetro de condição de parada ϵ_0, utilizamos $\epsilon_0 = \frac{\delta}{MelhorUB}$, onde $MelhorUB$ é uma constante que representa o melhor limite superior conhecido na lit-

eratura para a instância corrente.

Nosso método foi testado nas instâncias consideradas mais difíceis encontradas no OR-Libray. São elas: c10400, d10400, e10400, c20400, d20400, e20400, c40400, d40400, e40400, c15900, d15900, e15900, c30900, d30900, e30900, c60900, d60900, e60900, c201600, d201600, e201600, c401600, d401600, e401600, c801600, d801600, e801600.

Dividimos este capítulo em duas seções: uma para mostrar algumas estatísticas referentes à execução do nosso método e outra para mostrar a comparação entre o nosso método e o método de geração de colunas proposto por Pigatti et al. (2005)

10.1 Estatísticas referentes à execução do nosso método

Devido ao fato de estarmos considerando 3 valores diferentes para o parâmetro η e ao fato do método estar sendo testado em 27 instâncias diferentes, um número muito grande de resultados foi gerado. Sendo assim, são exibidos gráficos que ilustram o comportamento do nosso método apenas para instâncias d20400, c30900, e201600 pois além de possuírem tamanhos diversificados, resumem bem tal comportamento.

As figuras 11.1, 11.2 e 11.3 mostram gráficos onde estão representados o tempo necessário para se encontrar um determinado valor da norma $|| \nabla\widetilde{L}(\pi) ||$. Observando os gráficos, podemos notar que os três métodos apresentam comportamentos muito semelhantes enquanto $|| \nabla\widetilde{L}(\pi) || \geq 0,02$. Para se obter valores da norma $|| \nabla\widetilde{L}(\pi) ||$ próximos de $0,01$ percebemos que o método $1(\eta = 0,3)$ obteve um comportamento mais estável, isto é, até mesmo para instâncias que esse método teve um maior tempo computacional em relação a algum dos outros dois métodos, a diferença não foi tão acentuada.

Observando as figuras 11.4, 11.5 e 11.6, que mostram gráficos representando a relação entre o número mínimo de iterações internas para se obter um determinado valor da norma $|| \nabla\widetilde{L}(\pi) ||$, percebemos que para valores da norma $|| \nabla\widetilde{L}(\pi) ||$ não tão pequenos, percebemos um melhor desempenho do método $1(\eta = 0,3)$, porém a medida que o valor da norma $|| \nabla\widetilde{L}(\pi) ||$ decresce, percebemos que o número de iterações deste método cresce muito. Isto acontece porque para valores da norma $|| \nabla\widetilde{L}(\pi) ||$ pequenos necessariamente o valor do erro δ também é pequeno devido as condições de parada do método o que torna a função dual suave quase não-diferenciável. Nesse caso uma redução maior do erro δ pode fazer com que o método tenha maior dificuldade para encontrar a direção de

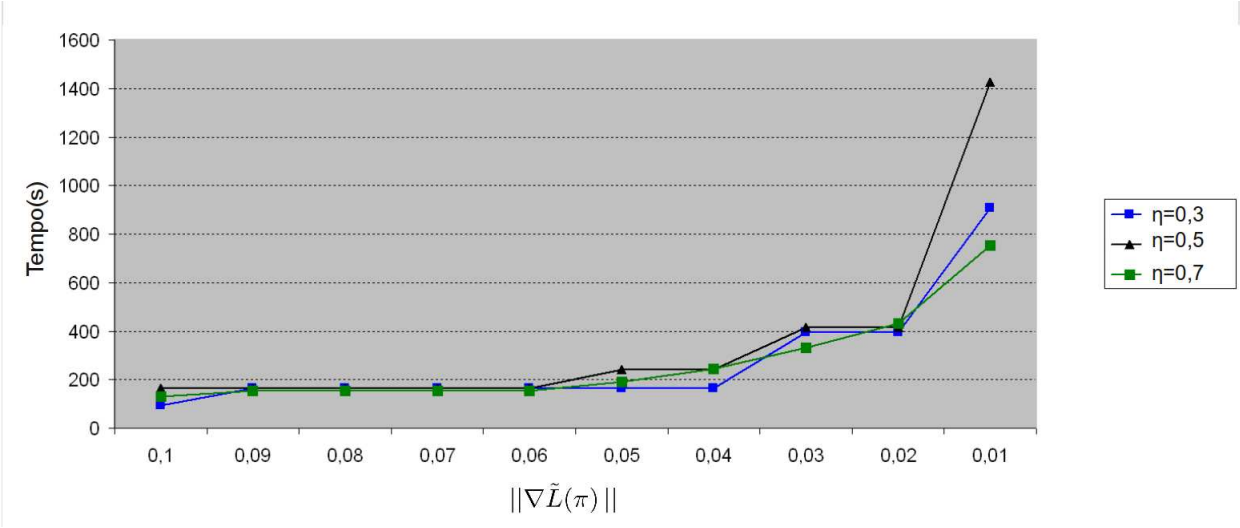

Figura 10.1: Instância d20400: gráfico $||\nabla\widetilde{L}(\pi)||$ × tempo

decrescimento da norma $||\nabla\widetilde{L}(\pi)||$.

Ao relacionar limites inferiores e número de iterações internas(ver figuras 11.7, 11.8 e 11.9), as estatísticas mostram que para a maioria das instâncias, enquanto o limite inferior é pelo menos uma unidade menor que o ótimo, os métodos 1,2 e 3 possuem comportamentos similares independente do tamanho da instância. No entanto, para se encontrar o limite inferior ótimo, de acordo com as três figuras podemos afirmar que nenhum método domina outro pois cada um dos métodos foi melhor em uma determinada instância.

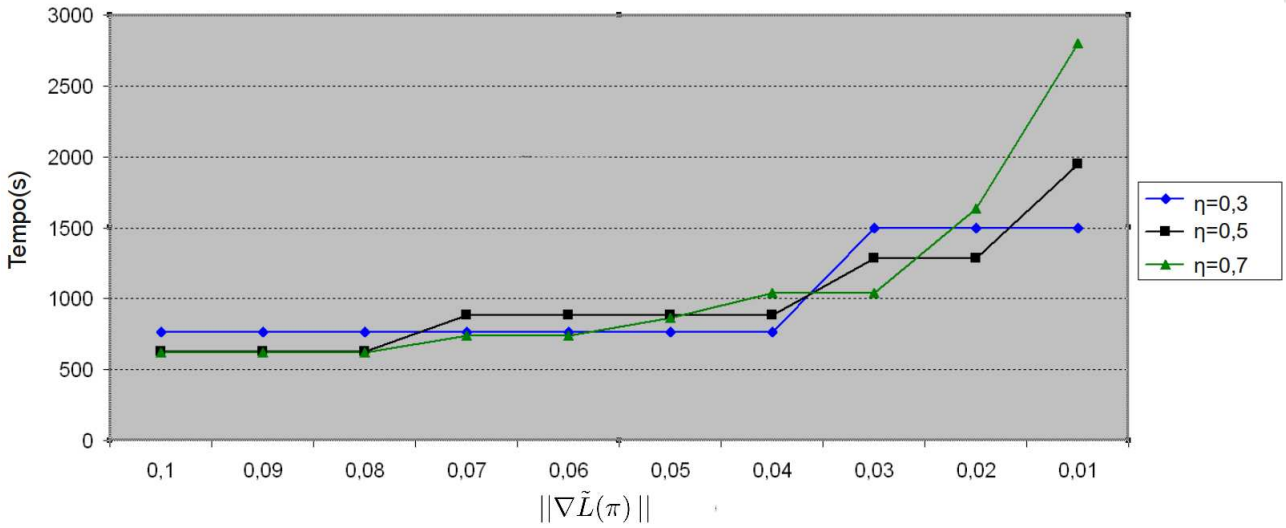

Figura 10.2: Instância c30900: gráfico $||\nabla\widetilde{L}(\pi)||$ × tempo

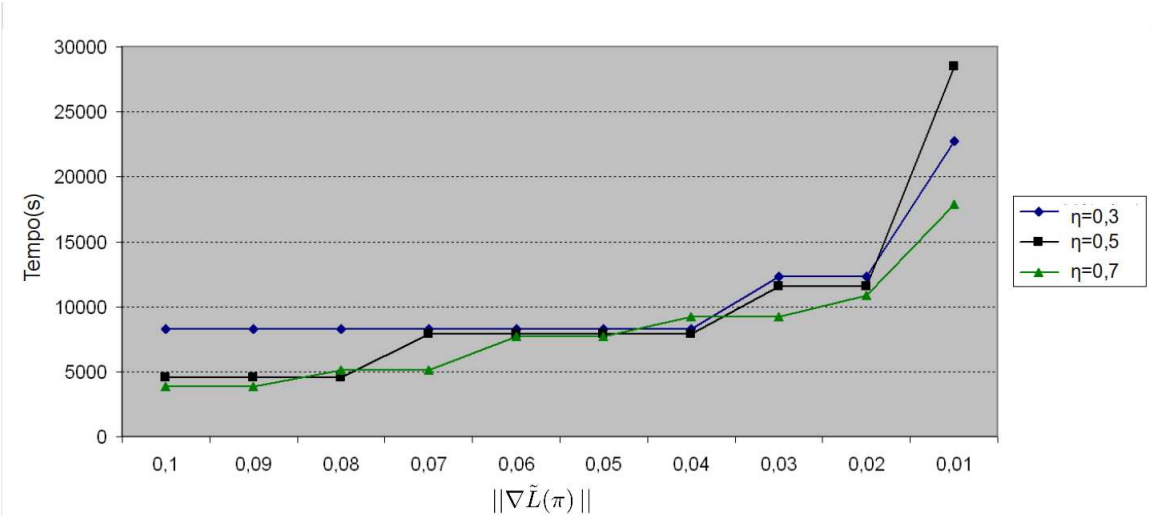

Figura 10.3: Instância e201600: gráfico $||\nabla\widetilde{L}(\pi)||$ × tempo

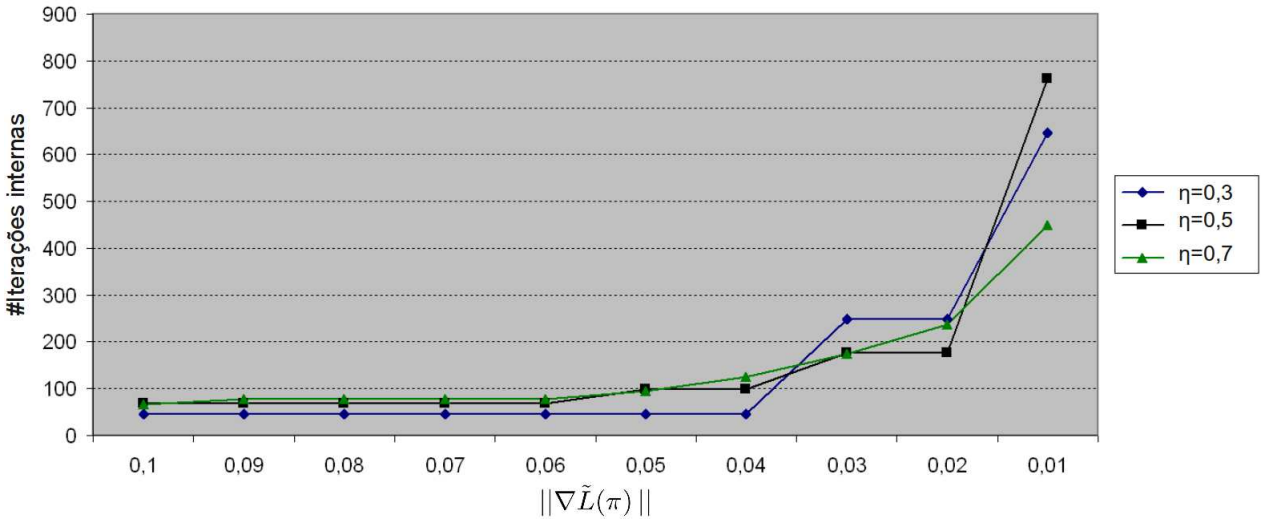

Figura 10.4: Instância d20400: gráfico $||\nabla \widetilde{L}(\pi)||$ × número de iterações

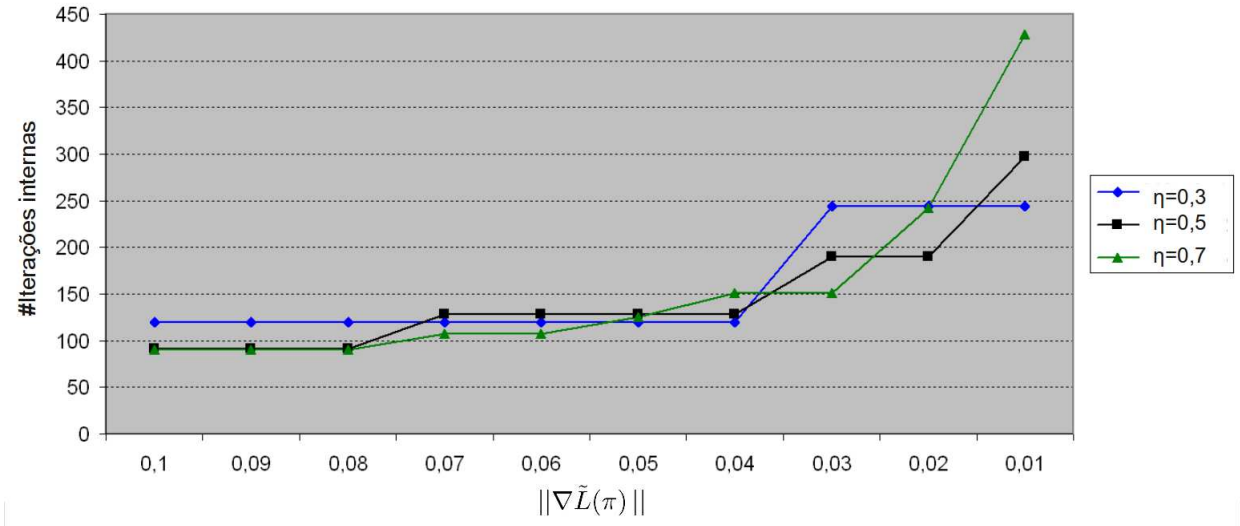

Figura 10.5: Instância c30900: gráfico $||\nabla \widetilde{L}(\pi)||$ × número de iterações

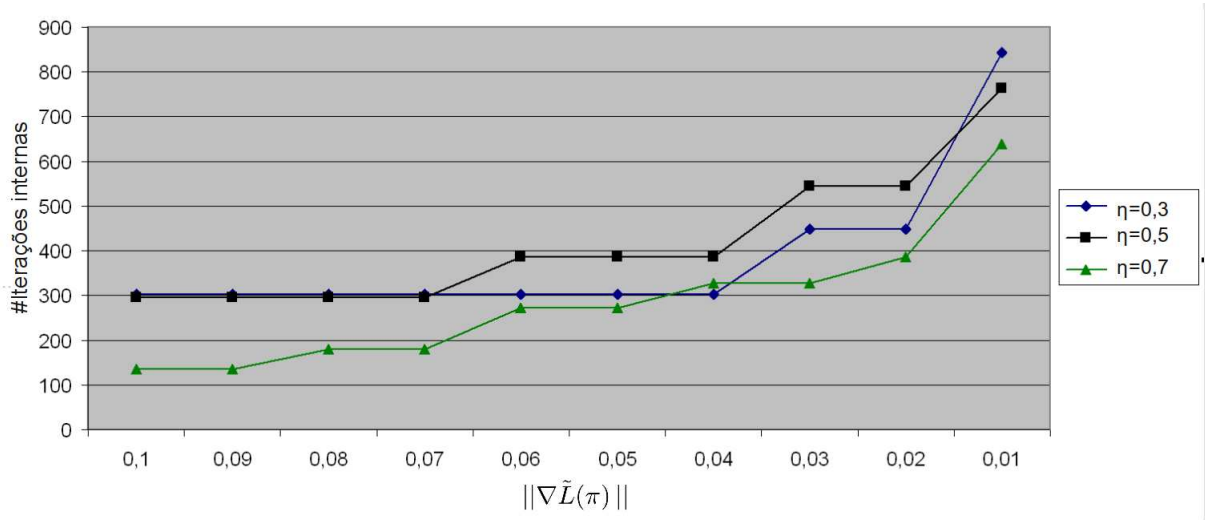

Figura 10.6: Instância e201600: gráfico $||\nabla \widetilde{L}(\pi)||$ × número de iterações

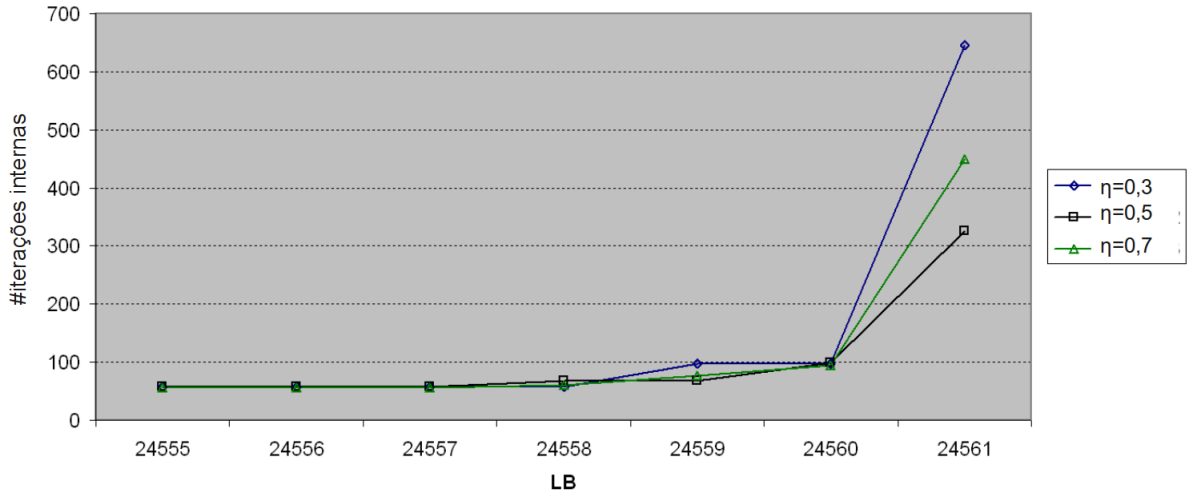

Figura 10.7: Instância d20400: gráfico Lower bound × número de iterações

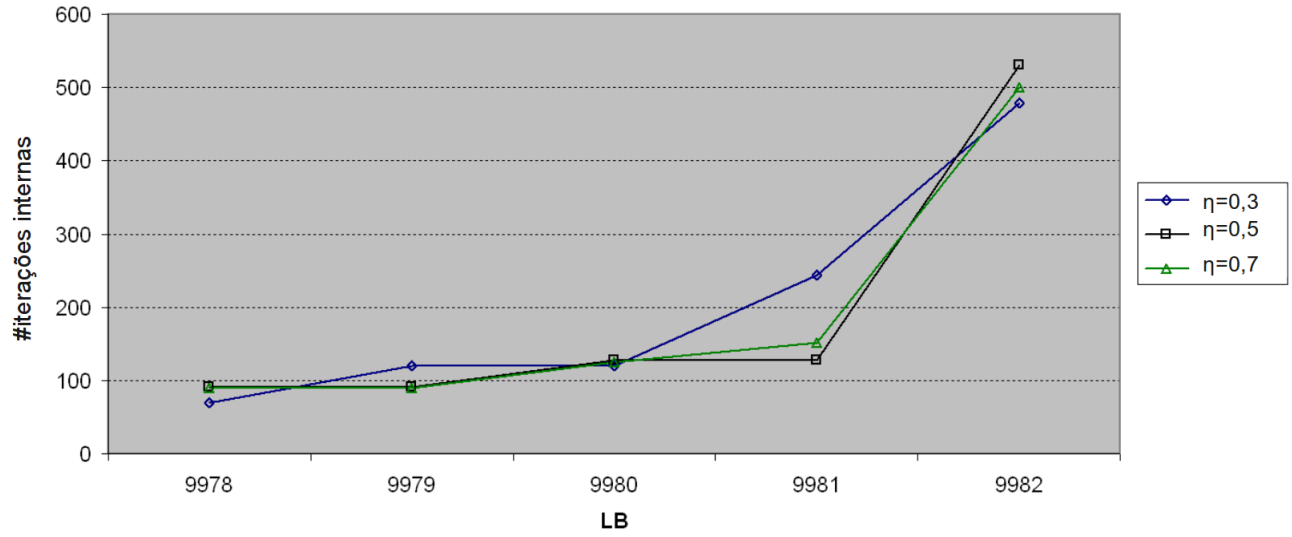

Figura 10.8: Instância c30900: gráfico Lower bound × número de iterações

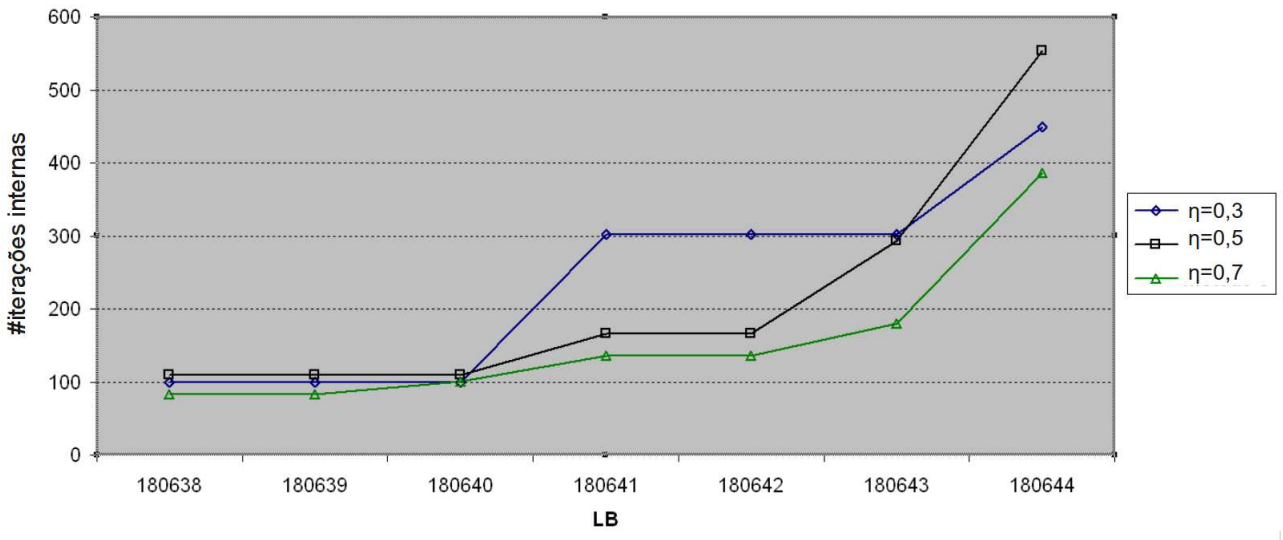

Figura 10.9: Instância e201600: gráfico Lower bound × número de iterações

10.2 Comparação do nosso método com o método de geração de colunas estabilizada

A tabela 11.1 mostra a comparação entre o nosso método, com três possíveis valores para η e o método proposto por Pigatti et al. (2005). O método de geração de colunas estabilizada utiliza as variáveis duais da solução ótima da formulação (6.1-6.4), substituindo 6.4 por $0 \leq x_{ij} \leq 1$ a relaxação de integrabilidade como um *hot start* para o método de geração de colunas. O tempo para resolver esta relaxação é desprezível quando comparado ao tempo total de execução do método. Para cada instância, nós marcamos em negrito o menor número de iterações e tempo. A legenda para essa tabela é a que segue:

- *Hot Start*: custo da relaxação linear da instância corrente.
- LB: Limite inferior obtido quando resolvendo subproblema lagrangeano original usando os multiplicadores de lagrange final.
- #Iter: Número de *minor iterations*.
- Tempo: Tempo de execução.

As instâncias que tiveram tempos de execução superiores a 100000 segundos foram abortadas. Nós decidimos não reportar os resultados para instâncias com $n = 16000$ pois o método de geração de colunas estabilizada para essas instâncias requer um enorme tempo de execução. Nós também colocamos em negrito o limite inferior 54552 obtido pelo método de geração de colunas estabilizada para a instância d60900 pois para nenhuma das variações do nosso método obtivemos este limite e sim um limite inferior uma unidade menor.

Observando a tabela 11.1, nós podemos reparar que o método proposto é mais rápido do que o método de geração de colunas estabilizada em 12 de 18 instâncias testadas tendo uma amplitude maior especialmente em instâncias com pequenos valores de $\frac{m}{n}$. Por exemplo, observemos as instâncias d10400, e10400, c15900, d15900, e15900 e e30900, onde o nosso método possui tempo de execução no mínimo 14 vezes menor. Observamos também as instâncias c10400 e c30900, onde nosso método foi pelo menos 4 vezes mais rápido. Isto ocorre porque a medida que $\frac{m}{n}$ decresce, o número de iterações requirida por

Pigatti et al. (2005) cresce. Por outro lado, para instâncias com maiores valores de $\frac{m}{n}$, o método obteve menores tempos computacionais.

Ao compararmos os números de iterações dos 4 métodos podemos observar que o nosso método obteve um menor número de iterações em 12 das 18 instâncias testadas e que em geral, o método mais rápido também possui um menor número de iterações para uma determinada instância.

Vale mencionar que, com o intuito de encontrarmos o limite ótimo para a instância d60900, executamos mais uma vez as três variações do nosso método para essa instância mudando as condições de parada, obtendo $|| \nabla \widetilde{L}(\pi) || < 0,0001$ e $\delta < 0,00001$. Ainda assim não encontramos o mesmo limite reportado por Pigatti et al. (2005). Tal fato nos leva a pensar que provavelmente ocorreu um erro de precisão para o método de geração de colunas estabilizada para essa instância.

Outra observação importante que podemos extrair da tabela 11.1 é que o nosso método que considera $\eta = 0,3$ obteve na me metade das instâncias um menor tempo de execução para se encontrar o limite ótimo.

Tabela 10.1: Comparação com geração de colunas estabilizada

	Hot	Este trabalho($\eta = 0,3$)			Este trabalho($\eta = 0,5$)			Este trabalho($\eta = 0,7$)			Pigatti et al.		
Instance	*Start*	LB	# Iter	Tempo	LB	# Iter.	Tempo	LB	# Iter	Tempo	LB	#Iter.	Tempo
c10400	5592	5596	**131**	**91,3**	5596	146	141,8	5596	158	120,5	5596	869	687,1
d10400	24956	24959	**79**	**138,4**	24959	98	244,1	24959	81	172,0	24959	8238	2043,3
e10400	45740	45745	220	**108,4**	45745	**194**	139,0	45745	197	109,1	45745	2222	1895,6
c20400	4775	4781	**332**	**202,1**	4781	380	328,4	4781	440	282,4	4781	623	872,9
d20400	24553	24561	646	906,5	24561	**325**	709,5	24561	450	751,6	24561	339	**158,2**
e20400	44862	44876	665	**312,0**	44876	582	400,5	44876	848	410,7	44876	**527**	479,3
c40400	4232	4244	251	146,8	4244	395	325,7	4244	307	195,4	4244	**176**	**91,9**
d40400	24348	24350	704	942,9	24350	763	1454,2	24350	829	1222,4	24350	**92**	**70,7**
e40400	44524	44557	844	393,2	44557	630	432,0	44557	835	405,3	44557	**134**	**37,2**
c15900	11337	11339	**106**	**674,3**	11339	113	812,4	11339	123	821,3	11339	5410	83898,2
d15900	55401	55403	188	**2040,0**	55403	**149**	2117,5	55403	188	2372,3	-	-	>100000
e15900	102417	102420	**258**	**1391,9**	102420	369	2077,3	102420	269	1505,6	-	-	>100000
c30900	9975	9982	479	2830,8	9982	**128**	**884,3**	9982	500	3250,8	9982	1020	7619,7
d30900	54829	54833	435	4568,9	54833	384	5074,2	54833	**350**	**4063,2**	54833	1160	11619,3
e30900	100413	100427	351	1864,2	100427	**250**	**1420,0**	100427	280	1550,6	100427	1518	67583
c60900	9316	9325	**349**	**2038,1**	9325	385	2470,5	9325	431	2609,6	9325	354	3256,8
d60900	54551	54551	275	2919,4	54551	334	4386,2	54551	404	4561,3	**54552**	**97**	**1417**
e60900	100103	100147	665	3451,6	100147	786	4349,1	100147	970	5154,5	100147	**380**	**1775**

Capítulo 11

Conclusões

Nesta dissertação, nós propomos um método para suavizar a função dual lagrangeana quando o subproblema é resolvido via programação dinâmica. Como um resultado, nós estamos aptos a encontrar boas aproximações do custo dual lagrangeano ótimo em tempos razoáveis. Nós também provamos que, quando o método converge para a solução ótima da função dual suavizada, uma solução fracionária viável para o problema primal é também fornecida. Nosso método permite uma convergência robusta, pelo menos para valores não muito pequenos de δ, pois aplica algoritmos de convergência quadrática. Isto é conseguido porque a suavização pode ser vista como um método que permite extrair informações a cerca de muitas soluções primais com uma única solução do subproblema lagrangeano. No entanto, este benefício é obtido ao custo de perda de generalidade pois o subproblema tem que ser resolvido por programação dinâmica.

Além disso, nós fornecemos experimentos e aplicamos a nossa técnica para o PAG e comparamos nossos resultados com o procedimento da geração de colunas proposto em (Pigatti et al., 2005). Os resultados reportados mostraram que nosso método se mostrou adequado para a maioria das instâncias fornecidas, especialmente quando a razão m/n é pequena. Finalmente, nós acreditamos que dentre os possíveis fatores negativos do nosso método, dois são evidentes: a existência de implementações muito eficientes para resolver subproblemas da mochila e o fato de estarmos transformando um problema linear em um problema quadrático. Devido a este fato, nós acreditamos que a técnica proposta nessa dissertação pode alcançar resultados ainda melhores para outros problemas de otimização combinatória onde a melhor maneira para resolver os subproblemas é através da programação dinâmica padrão e o problema dual é difícil para métodos de geração de

colunas(Rodrigues et al., 2008; Pessoa et al., 2008).

Referências Bibliográficas

F. Barahona e R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, 2000.

R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot e F. Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113(2): 299–344, 2008.

T.H. Cormen, C.E. Leiserson, R.L. Rivest e C. Stein. *Introduction to algorithms*, The MIT press, 2001.

T.S.A.O.C.D. DA, C. DOS, F. COMO, P. DOS REQUISITOS, N.P.A.O. DO GRAU, E.M.C. DE DOUTOR e E.M.E. MECÂNICA. MATRIZES QUASE-NEWTON ESPARSAS PARA PROBLEMAS DE OTIMIZAÇÃO NÃO LINEAR DE GRANDE PORTE.

RC Di Bello. *Análise do Comportamento da Umidade do Solo no Modelo Chuva-Vazão SMAP II–Versão com Suavização Hiperbólica. Estudo de Caso: Região de Barreiras na Bacia do rio Grande–BA*. Tese de doutoramento, Tese de Mestrado, Engenharia Civil, COPPE/UFRJ, Rio de Janeiro, 2005.

O. Du Merle, J.L. Goffin e J.P. Vial. On improvements to the analytic center cutting plane method. *Computational Optimization and Applications*, 11(1):37–52, 1998.

D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009, 1978.

L.G.A. Espejo e R.D. Galvão. O uso das relaxações lagrangeana e surrogate em problemas de programação inteira. *Pesquisa Operacional*, 22:387–402, 2002.

M.L. Fisher. Optimal solution of scheduling problems using Lagrange multipliers: Part I. *Operations Research*, p. 1114–1127, 1973.

M.L. Fisher. A dual algorithm for the one-machine scheduling problem. *Mathematical Programming*, 11(1):229–251, 1976.

M.L. Fisher. An applications oriented guide to Lagrangian relaxation. *Interfaces*, p. 10–21, 1985.

M.L. Fisher, R. Jaikumar e L.N. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, 1986. ISSN 0025-1909.

A. Friedlander. Elementos de programação não-linear. *Editora UNICAP*, 1994.

A. Friedlander e JM Martínez. New algorithms for maximization of concave functions with box constraints. *RAIRO. Recherche opérationnelle*, 26(3):209–236, 1992.

H. Fritzsche. Programação Não-Linear Análise e Métodos. *São Paulo, Editora Edgard Bl ”ucher*, 1977.

R.D. Galvao, L. Gonzalo Acosta Espejo e B. Boffey. A comparison of Lagrangean and surrogate relaxations for the maximal covering location problem. *European Journal of Operational Research*, 124(2):377–389, 2000.

A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2(2):82–114, 1974.

PC Gilmore e RE Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9(6):849–859, 1961.

PC Gilmore e RE Gomory. A linear programming approach to the cutting stock problem-part II. *Operations Research*, 11(6):863–888, 1963.

M. Guignard. Lagrangean relaxation. *Top*, 11(2):151–200, 2003.

M. Guignard e S. Kim. Lagrangean decomposition: A model yielding stronger lagrangean bounds. *Mathematical Programming*, 39(2):215–228, 1987. ISSN 0025-5610.

M. Guignard e M. Rosenwein. An improved dual-based algorithm for the generalized assignment problem. *Operations Research*, 37:658–663, 1989.

M. Held e R.M. Karp. The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1(1):6–25, 1971.

M.R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

M.R. Hestenes e E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand*, 49(6):409–436, 1952.

N. Karabakal, JC Bean e JR Lohmann. A steepest descent multiplier adjustment method for the generalized assignment problem. Report 92-11, University of Michigan. *Ann Arbor, MI*, 1992.

H. Kellerer, U. Pferschy e D. Pisinger. *Knapsack problems*, Springer, 2004.

N. Kohl e O.B.G. Madsen. An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research*, 45(3):395–406, 1997.

E.L. Lawler. *The traveling salesman problem: a guided tour of combinatorial optimization*, John Wiley & Sons Inc, 1985.

EL Lawler e DE Wood. Branch-and-bound methods: A survey. *Operations Research*, 14 (4):699–719, 1966.

D.G. Luenberger e Y. Ye. *Linear and nonlinear programming*, Springer Verlag, 2008.

RE Marsten, WW Hogan e JW Blankenship. The Boxstep method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.

J. C. Meza, P. D. Hough, P. J. Williams e R. A. Oliva. Opt++ documentation. Acessado em: 10 de junho de 2009., 2007. `http://software.sandia.gov/opt++`.

A. O. Moreno, A. E. Xavier e M. Souza. Um algoritmo de suavizaï¿½ï¿½o hiperbï¿½lica para resoluï¿½ï¿½o de problemas de classificaï¿½ï¿½o. *Simpï¿½sio de Pesquisa Operacional e Logï¿½stica da Marinha*, 2008.

A. Pessoa, P. Hahn, M. Guignard e Zhu Y-R. An improved algorithm for the generalized quadratic assignment problem. Em *Anais do XL Simpï¿½sio Brasileiro de Pesquisa Operacional*, 2008.

A. Pigatti, M.P. de Aragao e E. Uchoa. Stabilized branch-and-cut-and-price for the generalized assignment problem. Em *2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics, Electronic Notes in Discrete Mathematics*, 2005. p. 385–395.

M.L. Pinedo. *Scheduling: theory, algorithms, and systems*, Springer Verlag, 2008.

N. D. Pizzolato e A. A. Gandolpho. *Tï¿½cnicas de otimizaï¿½ï¿½o*, LTC, 2009.

G.M. Ribeiro e L.A.N. Lorena. Método de Geração de Colunas para o Problema do Carregamento de Paletes do Produtor. 2005.

R. Rodrigues, A. Pessoa, E. Uchoa e M. Poggi de Aragão. Algorithms over arc-time indexed formulations for single and parallel machine scheduling problems. Relatório Técnico RPEP Vol.8 no.8, Universidade Federal Fluminense, Engenharia de Produção, 2008.

A. Schrijver. *Theory of linear and integer programming*, John Wiley & Sons Inc, 1998.

E.L.F. Senne, L.A.N. Lorena e M.A. Pereira. Um algoritmo branch-and-price para problemas de localização de p-medianas. *Simpósio Brasileiro de Pesquisa Operacional*, 35, 2003.

T. Volgenant e R. Jonker. Branch and Bound Algorithm for the Symmetric Traveling Salesman Problem Based on the 1-Tree Relaxation. *EUROP. J. OPER. RES.*, 9(1): 83–89, 1982.

Apêndice A

Prova das propriedades referentes à função suave $\tilde{f}$

Teorema 10 *Para todo $\delta > 0$, $\tilde{f}$ é côncava.*

Prova: Para provar que $\tilde{f}$ é côncava, é suficiente provar que $g(z) = \sqrt{z^2 + 4\delta^2}$ é convexa. Isto é verdade se mostrarmos que $g''(z) > 0$ para todo z. Para isso, observemos que

$$g^{'}(z) = \frac{z}{\sqrt{z^2 + 4\delta^2}}$$

logo

$$\begin{aligned} g^{''}(z) &= -\frac{z^2}{\sqrt{(z^2 + 4\delta^2)^3}} + \frac{1}{\sqrt{z^2 + 4\delta^2}} \\ &= \frac{1}{\sqrt{z^2 + 4\delta^2}}(1 - \frac{z^2}{z^2 + 4\delta^2}) \end{aligned}$$

Como $\frac{1}{\sqrt{z^2+4\delta^2}}$ e $(1 - \frac{z^2}{z^2+4\delta^2})$ são positivos então $g''(z) > 0$ para todo z. ■

Teorema 11 *Para todo $\delta > 0$, a seguinte desigualdade é válida:*

$$f(x, y) - \delta < \tilde{f}(x, y) < f(x, y)$$

Prova: Suponhamos, sem perda de generalidade, que $x \leq y$, ou seja, $x - y \leq 0$ e $f(x, y) = x$.

Sabemos que, como $x - y \leq 0$,

$$\begin{aligned}
x - y &< \frac{3\delta}{4} \\
3\delta - 4(x-y) &> 0 \\
3\delta^2 - 4(x-y)\delta &> 0 \\
3\delta^2 - 4(x-y) + (x-y)^2\delta &> (x-y)^2 \\
4\delta^2 - 4(x-y) + (x-y)^2\delta &> (x-y)^2 + \delta^2 \\
(2\delta - (x-y))^2 &> (x-y)^2 + \delta^2 \\
2\delta - (x-y) &> \sqrt{(x-y)^2 + \delta^2} \\
-2\delta + (x-y) &< -\sqrt{(x-y)^2 + \delta^2} \\
-2\delta + 2x - x - y &< -\sqrt{(x-y)^2 + \delta^2} \\
-2\delta + 2x &< x + y - \sqrt{(x-y)^2 + \delta^2} \\
-\delta + x &< \frac{x + y - \sqrt{(x-y)^2 + \delta^2}}{2}
\end{aligned}$$

Como $x < y$, temos $f(x,y) = x$, e portanto a equação prévia é equivalente a

$$f(x,y) - \delta < \tilde{f}$$

Para provar a segunda parte da desigualdade devemos lembrar que

$$\begin{aligned}
\delta^2 &> 0 \\
\delta^2 + (x-y)^2 &> (x-y)^2 \\
\sqrt{\delta^2 + (x-y)^2} &> -(x-y) \\
-\sqrt{\delta^2 + (x-y)^2} &< (x-y) \\
x + y - \sqrt{\delta^2 + (x-y)^2} &< 2x \\
\frac{x + y - \sqrt{\delta^2 + (x-y)^2}}{2} &< x \\
\frac{x + y - \sqrt{\delta^2 + (x-y)^2}}{2} &< x = f(x,y) \quad \blacksquare
\end{aligned}$$

Teorema 12

$$\lim_{(x-y)\to+\infty} \tilde{f}(x,y) \;=\; y.$$

Prova:

$$\lim_{(x-y)\to+\infty} \tilde{f}(x,y) \;=\; \lim_{(x-y)\to+\infty} \frac{x+y-\sqrt{(x-y)^2+\delta^2}}{2} =$$

$$\lim_{(x-y)\to+\infty} y + \frac{x-y-\sqrt{(x-y)^2+\delta^2}}{2} \;=\; \lim_{(x-y)\to+\infty} y + \frac{-\delta^2}{2[(x-y)+\sqrt{(x-y)^2+\delta^2}]} = y \quad \blacksquare$$

Teorema 13

$$\lim_{(x-y)\to-\infty} \tilde{f}(x,y) \;=\; x.$$

Prova: A prova é análoga a do teorema prévio. ■

As outras três propriedades com relação a função $\tilde{f}$ possuem provas triviais e portanto serão omitidas deste trabalho. As três propridedades são :

- $\frac{\partial \tilde{f}(x,y)}{\partial x} = \frac{1}{2} - \frac{2(x-y)}{4\sqrt{(x-y)^2+4\delta^2}} = \frac{1}{2}\left(1 - \frac{x-y}{\sqrt{(x-y)^2+4\delta^2}}\right).$
- $x = y \Rightarrow \frac{\partial \tilde{f}(x,y)}{\partial x} = \frac{1}{2}.$
- $\frac{\partial \tilde{f}(x,y)}{\partial x} + \frac{\partial \tilde{f}(x,y)}{\partial y} = 1 - \left(\frac{x-y+y-x}{\sqrt{(x-y)^2+4\delta^2}}\right) = 1.$

Apêndice B

Prova do teorema 8

Prova: Para cada $j \in J$, basta provar que $\frac{\partial \widetilde{OPT_i}(n,b_i)}{\partial \pi_j} = x_{ij}$. Com isso, teremos $\forall j \in J$:

$$\begin{aligned}
\sum_{i=1}^{m} \frac{\partial \widetilde{OPT}(n,b_i)}{\partial \pi_j} &= \sum_{i=1}^{m} x_{ij}. \\
\sum_{i=1}^{m} \frac{\partial \widetilde{OPT}(n,b_i)}{\partial \pi_j} - 1 &= \sum_{i=1}^{m} x_{ij} - 1. \\
\sum_{i=1}^{m} \frac{\partial \widetilde{OPT}(n,b_i)}{\partial \pi_j} - \frac{\partial \sum_{j=1}^{n} \pi_j}{\partial \pi_j} &= \sum_{i=1}^{m} x_{ij} - 1. \\
\frac{\partial \widetilde{L}(\pi)}{\partial \pi_j} &= v(j). \\
\nabla \widetilde{L}(\pi) &= v.
\end{aligned}$$

Para tal prova, nós precisamos de 3 lemas. Antes da prova e apresentação de cada lema, nós daremos uma explicação intuitiva de cada um deles.

Para entendermos o lema 1, nós devemos lembrar a definição da matriz AUX_i mostrada no capítulo 8 . Por esta definição, o valor de cada célula $AUX_i(l,k)$ pode ser reescrito como uma soma de produtos de h's, onde cada produto é associado a um caminho que começa em $AUX_i(n,b_i)$ e termina em $AUX_i(l,b_i-k)$.

De agora em diante, para facilitar a notação, usamos AUX ao invés de AUX_i e $\widetilde{OPT}$ ao invés de $\widetilde{OPT_i}$. Também, nós usamos $\bar{Q}$ para denotar $\{j,...,n\}\backslash Q$ inside a sum over all $Q \in S^i_{j,k}$.

Lema 1 *Dado $i \in I$ e $k \in \{0,...,b_i\}$.$\forall j \in J$, nós temos:*

$$AUX(j, b_i - k) = \sum_{Q \in S^i_{j+1,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l}.$$

Prova: Antes da prova, para ilustrar o teorema anterior, calculamos o valor de $AUX_1(1,3)$, onde $k = 7$, $i = 1$ e $j = 1$. Neste caso nós temos $S^1_{2,7} = \{\{2,4\},\{3\}\}$:

$$\begin{aligned} \sum_{Q \in S^1_{2,7}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} &= \prod_{l \in \{2,4\}} h^{1,k_1(\{2,4\},l)}_{y,l} \times h^{1,k_1(\{2,4\},3)}_{x,3} + h^{1,k_1(3)}_{y,3} \times \prod_{l \in \{2,4\}} h^{1,k_1(3)}_{x,l} \\ &\cong 0,9796 \end{aligned}$$

Agora, nós provamos o lema por indução sobre o valor de j, onde $j = n$ será a base da indução. Nós devemos lembrar que:

$$AUX_i(n, b_i - k) = \begin{cases} 1 & se\, k = 0 \\ 0 & se\, k \neq 0 \end{cases},$$

Para $j = n$ nós dividiremos a prova em dois casos:

caso 1: $k = 0$.

Por definição, $S^i_{n+1,0} = \{\emptyset\}$. Portanto $\sum_{Q \in S^i_{j+1,0}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} = 1$.

casoe 2: $k \neq 0$.

$S^i_{n+1,k} = \emptyset$. Assim, $\sum_{Q \in S^i_{j+1,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(\bar{Q},l)}_{x,l} = 0$.
Assim, a proposição é válida para $j = n$.

Suponhamos que a proposição é válida para um dado j_0, onde $j_0 \in \{2, ..., n\}$. Agora, nós provaremos que a proposição também é válida para $j = j_0 - 1$:

$$AUX(j_0 - 1, b_i - k) = h^{i,k}_{x,j_0} \times AUX(j_0, b_i - k) + h^{i,k}_{y,j_0} \times AUX(j_0, b_i - k + a_{ij_0})$$

Pela hipótese de indução, a expressão acima é equivalente a

$$h_{x,j_0}^{i,a_{ij_0}} \times \sum_{Q \in S^i_{j_0+1,k}} \left(\prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \right) + h_{y,j_0}^{i,a_{ij_0}} \times \sum_{Q \in S^i_{j_0+1,k-a_{ij_0}}} \left(\prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \right) =$$

$$\sum_{Q \in S^i_{j_0,k}} \prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \quad \blacksquare$$

O próximo lema ajuda a mostrar como expandir cada componente do gradiente $\nabla \widetilde{OPT}(n, b_i)$ em uma soma de produtos de h's assim como nós já fizemos com a matriz AUX. Nós apresentaremos tal expansão no lema 3.

Lema 2 *Dado $i \in M$ e $q \in J - \{n\}$.$\forall j \leq q$, nós temos :*

$$\sum_{k=0}^{b_i} \sum_{Q \in S^i_{q+1,k}} \prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \times \frac{\partial \widetilde{OPT}(q, b_i - k) + |Q \bigcap \{q+1\}| \times \pi_{q+1}}{\partial \pi_j} =$$

$$\sum_{k=0}^{b_i} \sum_{Q \in S^i_{q,k}} \prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \times \frac{\partial \widetilde{OPT}(q-1, b_i - k) + |Q \bigcap \{q\}| \times \pi_q}{\partial \pi_j}.$$

Prova: Aplicando a recorrência (7.8) em

$$\sum_{k=0}^{b_i} \sum_{Q \in S^i_{q+1,k}} \prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \times \frac{\partial \widetilde{OPT}(q, b_i - k) + |Q \bigcap \{q+1\}| \times \pi_{q+1}}{\partial \pi_j} =$$

nós obtemos:

$$= \sum_{k=0}^{b_i} \sum_{Q \in S^i_{q+1,k}} \prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \times \frac{\partial \tilde{f}(\widetilde{OPT}(q-1, b_i - k) + 0, \widetilde{OPT}(q-1, b_i - k - a_{iq}) + \pi_q) + |Q \bigcap \{q+1\}| \times \pi_{q+1}}{\partial \pi_j} =$$

$$= \sum_{k=0}^{b_i} \sum_{Q \in S^i_{q+1,k}} \prod_{l \in Q} h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \times (h_{x,q}^{i,k} \times \frac{\partial \widetilde{OPT}(q-1, b_i - k) + 0}{\partial \pi_j} + h_{y,q}^{i,k+a_{il}} \times \frac{\partial \widetilde{OPT}(q-1, b_i - k - a_{iq-1}) + \pi_q}{\partial \pi_j}),$$

A expressão é equivalente a:

$$\sum_{k=0}^{b_i} \sum_{Q \in S^i_{q,k}} \prod h_{y,l}^{i,k_i(Q,l)} \times \prod_{l \in \bar{Q}} h_{x,l}^{i,k_i(Q,l)} \times \frac{\partial \widetilde{OPT}(q-1, b_i - k) + |Q \bigcap \{q\}| \times \pi_q}{\partial \pi_j},$$

porque, para cada $Q \in S^i_{q+1,k}$, os dois termos no interior da soma representa os dois subconjuntos $Q^{'} \in S^i_{q,k}$ que Q: o próprio Q ($q \in \bar{Q}^{'}$) e $Q \cup \{q\}$. ■

Como já mencionado, O lema 3 expande cada componente do gradiente $\nabla\widetilde{OPT}(n, b_i)$ em uma soma de produtos de h's.

Lema 3 *Seja $i \in M$ e $j \in J$.$\forall q \in \{1, ..., n\}$, nós temos:*

$$\frac{\partial\widetilde{OPT}(n, b_i)}{\partial\pi_j} = \sum_{k=0}^{b_i} \sum_{Q \in S^i_{q,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} \times \frac{\partial\widetilde{OPT}(q-1, b_i - k) + |Q \bigcap \{q\}| \times \pi_q}{\partial\pi_j}$$

Prova: Nós provamos este lema por indução sobre o valor de q. Para $q = n$, nós temos

$$\frac{\partial\widetilde{OPT}(n, b_i)}{\partial\pi_j} = \frac{\partial\tilde{f}(\widetilde{OPT}(n-1, b_i), \widetilde{OPT}(n-1, b_i - a_{in}) + c_{in} + \pi_n)}{\partial\pi_j} =$$

$$= h^{i,0}_{x,n} \times \frac{\partial\widetilde{OPT}(n-1, b_i)}{\partial\pi_j} + h^{i,a_{in}}_{y,n} \times \frac{\partial\widetilde{OPT}(n-1, b_i - a_{in}) + c_{in} + \pi_n}{\partial\pi_j},$$

que é consistente com este lema porque os únicos valores de k para que $S^i_{n,k} \neq \emptyset$ são $k = 0$ e $k = a_{in}$.

Agora, suponhamos que este lema é válido para todo $q > q_0$. Para $q = q_1$, nós temos

$$\frac{\partial\widetilde{OPT}(n, b_i)}{\partial\pi_j} = \sum_{k=0}^{b_i} \sum_{Q \in S^i_{q+1,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} \times \frac{\partial\widetilde{OPT}(q, b_i - k) + |Q \bigcap \{q+1\}| \times \pi_{q+1}}{\partial\pi_j}. (I)$$

Pelo lema 2, o lado direito de (I) é equivalente a:

$$\sum_{k=0}^{b_i} \sum_{Q \in S^i_{q,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} \times \frac{\partial\widetilde{OPT}(q-1, b_i - k) + |Q \bigcap \{q\}| \times \pi_q}{\partial\pi_j} \quad ■$$

Com intuito de provar este teorema, nós aplicamos a expressão dada pelo lema 1 na definição de x_{ij} dada por (8.1). Então, nós obtemos $x_{ij} = \sum_{k=0}^{b_i} h^{i,k}_{y,j} \sum_{Q \in S^i_{j+1,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l}$.

Pelo Lema 3, para $q = j + 1$, e pela definição de $\widetilde{OPT}$ mostrada em (7.8) nós obtemos:

$$\frac{\partial \widetilde{OPT}(n, b_i)}{\partial \pi_j} =$$

$$= \sum_{k=0}^{b_i} \sum_{Q \in S^i_{j+1,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} \times \frac{\partial \tilde{f}(\widetilde{OPT}(j-1, b_i - k) + 0, \widetilde{OPT}(j-1, b_i - k - a_{ij}) + \pi_j) + |Q \bigcap \{j+1\}| \times \pi_{j+1}}{\partial \pi_j} =$$

$$= \sum_{k=0}^{b_i} \sum_{Q \in S^i_{j+1,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} \times \left(h^{i,k}_{x,j} \frac{\partial \widetilde{OPT}(j-1, b_i - k)}{\partial \pi_j} + h^{i,k}_{y,j} \times \left(\frac{\partial \widetilde{OPT}(j-1, b_i - k - a_{ij})}{\partial \pi_j} + \frac{\partial \pi_j}{\partial \pi_j} \right) \right) .$$

Como $\frac{\partial \widetilde{OPT}(j-1, b_i - k)}{\partial \pi_j} = 0, \frac{\partial \widetilde{OPT}(j-1, b_i - k - a_{ij})}{\partial \pi_j} = 0$ e $\frac{\partial \pi_j}{\partial \pi_j} = 1$, nós obtemos

$$\frac{\partial \widetilde{OPT}(n, b_i)}{\partial \pi_j} = \sum_{k=0}^{b_i} \sum_{Q \in S^i_{j+1,k}} \prod_{l \in Q} h^{i,k_i(Q,l)}_{y,l} \times \prod_{l \in \bar{Q}} h^{i,k_i(Q,l)}_{x,l} \times h^{i,k}_{y,j} = x_{ij} \quad \blacksquare.$$

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo