

Luiza Amalia Pinto

**Métodos de Pontos Interiores para
Programação Inteira**



1910000863



Universidade Estadual Paulista

Júlio de Mesquita Filho

Métodos de Pontos Interiores para Programação Inteira

Métodos de Pontos Interiores Para Programação Inteira

Luiza Amalia Pinto

**Dissertação de Mestrado
Pós-Graduação em Matemática Aplicada
MAP-041**

São José do Rio Preto, fevereiro de 1999.



Universidade Estadual Paulista

Júlio de Mesquita Filho

Métodos de Pontos Interiores para Programação Inteira

Luiza Amalia Pinto

Orientadora: Profa. Dra. Maria do Socorro Nogueira
Rangel



863/99

Dissertação apresentada ao Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto, como parte dos requisitos para a obtenção do título de Mestre em Matemática Aplicada .

São José do Rio Preto, fevereiro de 1999.



Vê esta mulher?

Entregando em tua casa, não me desse água para lavar os pés.

Esto, porém, regou meus pés com lágrimas e os enxugou com seus cabelos.

Não me beijastes ela, entregando, desde que entret não cessa de me beijar os pés.

Não me ungiste a cabeça com óleo, mas está com bálsamo ungida. À minha Mãe,

Por isso te digo: perdoadas lhe são as suas. Dona Nelcinda e ao

príncipe da realidade aqui, mas aquele a quem perdoas. meu namorado, Renato,

Lucas, 7: 44 dedico.

Agradecimentos

A prof. Dr.ª Maria de Siqueira Rangel, pela orientação e a amizade dedicada durante o mestrado. A todos os professores do DECE que contribuíram para a minha formação. Ao prof. Dr. Antonio Carlos Mineiro, da UNICAMP – IMECC, e ao prof. Dr. Carlos Eduardo Ferreira, da USP – IME, pela indicação de boas referências bibliográficas. A prof.ª e amiga Edma Cássia Baptista, da UNESP – Bauré, pelo carinho e incentivo. Ao pessoal da Secretaria de pós-graduação, e a todos os funcionários deste Instituto.

A minha Mãe, Dona Nelgunda, que sempre soube respeitar as minhas decisões e as iniciativas, pela confiança nos momentos difíceis e por me lembrar, inúmeras vezes, que é preciso continuar apesar de todas as dificuldades. Acho que quando DEUS escolheu um presente para me dar, escolheu a minha Mãe. Ao meu Pai, Miguel (in memoriam), que mesmo ausente esteve presente em muitas lembranças. Ao meu irmão Márcio, mesmo achando que já era hora de parar de estudar, me levou diversas vezes, a rodoviária de SP e me deixou na Ponte do Piquiri, obrigada pela carona, e pelo carinho de irmão. Ao Renato, companheiro, amigo e namorado, por tudo que pôde fazer por mim durante todo o tempo de mestrado, seja como ajuda pessoal ou profissional, sempre esteve ao meu lado incentivando, consolando e namorando (como não poderia deixar de ser). E por diversas vezes que fez este trabalho na tentativa de corrigir os erros de “plágio” e na elaboração da parte gráfica.

A amiga Inês, aos seus filhos Eduardo e Gabriel, ao seu marido José Luis e ao meu primo Leonel, pela amizade, carinho e hospitalidade oferecida a mim e à minha irmã Tereza.

Vê esta mulher ?

Entretanto em tua casa, não me deste água para lavar os pés;

Esta, porém, regou meus pés com lágrimas e os enxugou com seus cabelos.

Não me beijastes; ela, entretanto, desde que entrei não cessa de me beijar os pés.

Não me ungiste a cabeça com óleo, mas esta com bálsamo ungiu os meus pés.

*Por isso te digo: perdoados lhe são os seus muitos pecados,
porque ela muito amou, mas aquele a quem pouco se perdoa, pouco ama.*

Lucas, 7; 44 – 47

Agradecimentos

À prof^a. Dr^a. Maria do Socorro Rangel, pela orientação e a amizade dedicada durante o mestrado. A todos os professores do DCCE que contribuíram para a minha formação. Ao prof. Dr. Antonio Carlos Moretti, da UNICAMP – IMECC, e ao prof. Dr. Carlos Eduardo Ferreira, da USP – IME, pela indicação de boas referências bibliográficas. À prof^a. e amiga Edmea Cássia Baptista, da UNESP – Bauru, pelo carinho e incentivo. Ao pessoal da Secretaria de pós-graduação, e a todos os funcionários deste Instituto.

A minha Mãe, Dona Nelcinda, que sempre soube respeitar as minhas decisões e as incentivou, pelo conforto nos momentos difíceis e por me lembrar, inúmeras vezes, que é preciso continuar apesar de todas as dificuldades. Acho que quando *DEUS* escolheu um presente para me dar, escolheu a minha Mãe. Ao meu Pai, Miguel (*in memorian*), que mesmo ausente, esteve presente em minhas lembranças. Ao meu irmão, Márcio, mesmo achando que já era hora de parar de estudar, me levou diversas vezes a rodoviária de SP e me buscou na Ponte do Piqueri, obrigada pela carona, e pelo carinho de irmão. Ao Renato, companheiro, amigo e namorado, por tudo que pôde fazer por mim durante todo o tempo de mestrado, seja como ajuda pessoal ou profissional, sempre esteve ao meu lado incentivando, consolando e namorando (como não poderia deixar de ser). E pela diversas vezes que leu este trabalho na tentativa de corrigir os erros de “plural” e na elaboração da parte gráfica.

À amiga Inês, aos seus filhos Eduardo e Gabriel, ao seu marido José Luiz e ao meu primo Leonel, pela amizade, carinho e a hospedagem oferecida a mim e ao Renato. À irmã Terezinha e a todos os colegas do pensionato, obrigada pelo carinho.

À amiga Claudineia, pelas inúmeras vezes que me recebeu em sua casa, junto com seus familiares. Obrigada pela amizade e carinho de todos. E, na tentativa de não omitir nenhum nome, a todos os colegas da pós-graduação, pelo companheirismo e a todos que me acolheram em Rio Preto.

A *DEUS*, por sempre me guiar e iluminar o meu caminho.

Resumo

Os métodos clássicos de Programação Inteira usam o método Simplex para a resolução das relaxações lineares associadas. Nesta dissertação é abordado o uso dos métodos de Pontos Interiores na solução da relaxação, buscando uma alternativa ao método Simplex.

Primeiramente são apresentados os métodos de Pontos Interiores: Redução de Potencial e Trajetória Central. Num segundo momento são discutidas técnicas de adaptação destes algoritmos para os métodos de solução de Programação Inteira, em substituição ao Simplex. Por fim, abordamos uma aplicação do método de Planos de Cortes com Pontos Interiores para resolver o Problema de Ordenamento Linear.

Abstract

The classic methods of Integer Programming use the Simplex method for the resolution of the linear relaxations that arise. In this dissertation we present Interior Point methods as an alternative to Simplex method.

First of all we introduce Interior Point methods: Potential-Reduction and Path-Following. Afterwards we present techniques to adapt these algorithms for solving the linear relaxations that arise in the Cutting Plane and Branch and Cut methods. Finally, we consider an application of Cutting Plane methods with Interior Point to solve the Linear Ordering Problem.



Capítulo 1 – Introdução	1
-------------------------------	---

Capítulo 2 – Algoritmos de Pontos Interiores.....	3
---	---

2.1 Idéias de Programação Linear e Notações	4
---	---

2.2 Método Primal-Dual	6
------------------------------	---

2.2.1 A Trajetória Central	9
----------------------------------	---

2.2.2 Uma Estrutura Primal-Dual	12
---------------------------------------	----

2.2.3 Método de Pontos Interiores para Variáveis Canalizadas	13
--	----

2.3 Método de Redução Potencial (RP)	18
--	----

2.4 Método da Trajetória Central.....	22
---------------------------------------	----

2.4.1 Método da Trajetória Central de Passo Curto	23
---	----

2.4.2 Método Preditor-Corretor (PC)	24
---	----

2.4.3 Método da Trajetória Central de Passo Longo	26
---	----

2.4.4 Método de Pontos Interiores Inviável	27
--	----

2.5 Comentários Práticos sobre o Algoritmo de Pontos Interiores Primal-Dual	29
---	----

2.5.1 Solução Inicial	30
-----------------------------	----

2.5.2 Critério de Parada	33
--------------------------------	----

2.5.3 Algoritmo Primal-Dual	33
-----------------------------------	----



Capítulo 3 – Planos de Cortes e Pontos Interiores.....	35
3.1 Passos Iniciais	35
3.2 Gerando Planos de Cortes de Gomory	37
3.3 Planos de Cortes Gerados a parti de Pontos Interiores	39
3.4 Reotimização	44
3.5 Planos de Cortes com Pontos Interiores	47
Capítulo 4 – Branch and Cut e Método de Pontos Interiores	49
4.1 O Método Branch and Bound	50
4.2 O Método Branch and Cut	54
4.3 Uso de Pontos Interiores no Método Branch and Bound.....	56
Capítulo 5 – O Problema de Ordenamento Linear.....	63
5.1 Formulação do Problema	63
5.2 O Método de Planos de Cortes	73
Capítulo 6 – Conclusões	81
Bibliografia	83

Os métodos clássicos na resolução de problemas de Programação Inteira são: o Método de Planos de Cortes, o Método Branch-and-Bound e o Método Branch-and-Cut (que é o "casamento" dos métodos de Planos de Cortes e Branch-and-Bound). A ideia básica destes métodos consiste em relaxar de (PI) a restrição de integralidade e resolver o problema de Programação Linear (PL) resultante, chamado de *relaxação linear*. Se a solução da relaxação linear for inteira o (PI) está resolvido. Caso contrário a relaxação linear é iterativamente melhorada na busca da solução ótima do (PI). Assim, junto com as técnicas de cada método particular de (PI), são usadas também técnicas de (PL) para resolver o problema.

A pedra fundamental da Programação Linear foi lançada em 1947, por G. B. Dantzig, que desenvolveu o Método Simplex, o qual prevaleceu até 1979, quando L. G. Khachiyan



Capítulo

1

Introdução

Problemas de Programação Linear (PL) são problemas de maximização ou minimização de uma função linear, denominada *função objetivo*, sujeitos a um conjunto de equações ou inequações lineares, denominado *conjunto de restrições*, onde as variáveis usualmente devem ter valores não-negativos. Quando estas variáveis devem ter valores inteiros, os problemas levam o nome de *Problemas de Programação Inteira* (PI). Os (PI) são chamados de *puros* quando todas as variáveis devem ser inteiras, e de *mistos* quando apenas algumas variáveis são inteiras. E mais, quando as variáveis do problema *puro* ou *misto* estão restritas a tomarem valores 0 ou 1, chamamos de problemas *puro zero-um* (0-1) ou *zero-um e misto zero-um*, respectivamente. Neste trabalho abordaremos o Problema de Programação Inteira Puro, ou simplesmente Problema de Programação Inteira.

Os problemas de Programação Inteira têm uma grande variedade de aplicações, tais como: distribuição de tarefas [HP - 1993], planejamento de produção, problemas de armazenamento (estoques), entre outros. Sua aplicação se estende a áreas de Combinatória, Teoria dos Grafos e Lógica [Rangel - 1995].

Os métodos clássicos na resolução de problemas de Programação Inteira são: o *Método de Planos de Cortes*, o *Método Branch-and-Bound* e o *Método Branch-and-Cut* (que é o “casamento” dos métodos de Planos de Cortes e Branch-and-Bound). A idéia básica destes métodos consiste em retirar de (PI) a restrição de integralidade e resolver o problema de Programação Linear (PL) resultante, chamado de *relaxação linear*. Se a solução da relaxação linear for inteira o (PI) está resolvido. Caso contrário a relaxação linear é iterativamente *melhorada* na busca da solução ótima do (PI). Assim, junto com as técnicas de cada método particular de (PI), são usadas também técnicas de (PL) para resolver o problema.

A pedra fundamental da Programação Linear foi lançada em 1947, por G. B. Dantzig, que desenvolveu o *Método Simplex*, o qual prevaleceu até 1979, quando L. G. Khachiyan



propôs o Método dos Elipsóides para (PL) [Matumoto - 1996]. Este método mostrava que (PL) pode ser resolvido com um número de operações limitadas por um polinômio relacionado ao *tamanho* do problema. Entretanto, era ineficiente na prática, apesar de apresentar boas propriedades teóricas [Gonzaga - 1989].

Em 1984, N. Karmarkar publicou um algoritmo cuja complexidade teórica era bem melhor que a do método dos Elipsóides e, além disso, obteve resultados computacionais melhores que os obtidos com o método Simplex [Gonzaga - 1989]. A partir daí houve intensa pesquisa nesta área surgindo uma nova classe de métodos, conhecida como Métodos de Pontos Interiores. Na década seguinte, surgiram os Métodos Primais-Duais considerados os mais importantes desta classe, mostrando excelentes propriedades teóricas, boa performance prática e fácil compreensão de suas idéias fundamentais [Wright - 1997].

Os métodos de (PI) foram originalmente desenvolvidos utilizando o método Simplex para resolver as relaxações lineares envolvidas. A proposta deste trabalho é a de investigar o uso do método Primal-Dual como alternativa ao método Simplex. A principal dificuldade a ser resolvida quando se usa um Método de Ponto Interior em (PI) é a utilização da solução ótima de um problema linear como ponto de partida para a resolução de um problema relacionado. Em contraste com o método Simplex, Pontos Interiores pode usar informações do ponto corrente antes de atingida a otimalidade, e a sua reotimização é possível usando um ponto de partida não-ótimo. Este fato é o que possivelmente o tornará competitivo em relação ao método Simplex. Suporemos que o leitor já tenha uma certa intimidade com o método Simplex. Boas referências para este tópico podem ser encontradas em [BJ - 1977] e [Luenberger - 1989].

No capítulo 2 apresentamos o Método Primal-Dual para (PL), focalizando algumas variantes desta classe. No capítulo 3 abordamos o Método dos Planos de Cortes utilizando o Método Primal-Dual para a resolução dos problemas de (PL) associados. No capítulo 4 focalizamos o Método Branch-and-Bound e Branch-and-Cut, usando o Método Primal-Dual para resolver os subproblemas envolvidos. No capítulo 5 apresenta-se uma aplicação do Método de Planos de Cortes usando o método Primal-Dual, para resolver o problema de Ordenação Linear. Finalmente, apresentamos as conclusões (capítulo 6) e as referências usadas neste trabalho.



Capítulo 2

Métodos de Pontos Interiores

Até há pouco tempo problemas de Programação Linear eram resolvidos pelo *Método Simplex*, que foi introduzido por Dantzig em 1947. Esse quadro se alterou em 1984 quando Karmarkar publicou um artigo que abriu uma nova fronteira na resolução de (PL) [Gonzaga - 1989].

Karmarkar propôs um método que busca a solução do (PL) através do *interior* da região viável, ao contrário do Método Simplex que o faz pela fronteira. Karmarkar também provou que a complexidade de seu *Algoritmo Projetivo*, como era chamado, era polinomial, portanto melhor que a do Simplex, porém sua implementação não estava disponível [Matumoto - 1996].

Karmarkar abriu um novo campo de pesquisa e, a partir daí, houve muitas modificações no método, permanecendo contudo a sua idéia básica: a de caminhar pelo interior da região viável. Atualmente existem três classes de métodos: os que consideram apenas problemas no formato primal, outros apenas no formato dual e, finalmente métodos que extraem informação dos dois problemas, conhecidos como *Métodos Primais-Duais*.

Neste capítulo faremos uma breve revisão dos métodos Primais-Duais devido a intensidade com que estes vêm sendo tratados nos últimos anos e a sua aplicação em problemas de Programação Inteira. No entanto, omitiremos a discussão sobre a convergência e complexidade dos métodos apresentados. Sugerimos que o leitor interessado recorra às referências citadas ao longo do texto.



2.1 Idéias de Programação Linear e Notações

A maior parte do desenvolvimento teórico de algoritmos para (PL) é feita para problemas com restrições do tipo igualdade, conhecidos como problemas no *formato primal* ou no *formato padrão*. (PL) também pode ser formulado com restrições de desigualdade, que se chama *formato dual*. Os dois problemas são relacionados por expressões simples, que permitem traduzir métodos desenvolvidos em um formato para o outro [Gonzaga - 1989].

Considere o problema no formato primal (ou padrão):

$$\begin{array}{ll} \text{Min.} & c^T x \\ \text{(PL)} & \text{S. a } Ax = b, \\ & x \geq 0 \end{array}$$

onde $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ e $x, c \in \mathbb{R}^n$.

Suporemos que a matriz de restrições A tenha posto completo. O conjunto de soluções viáveis do problema (PL) é definido por:

$$P = \{x \in \mathbb{R}^n / Ax = b, x \geq 0\}.$$

O interior de P definido por:

$$P^0 = \{x \in \mathbb{R}^n / Ax = b, x > 0\},$$

é o conjunto das soluções interiores de (PL). Suporemos $P^0 \neq \emptyset$.

Todo ponto satisfazendo $x \in P^0$ será denominado de *ponto interior*.

Associado ao problema primal temos o problema dual, que consiste basicamente dos mesmos dados organizados de maneira diferente:

$$\begin{array}{ll} \text{Max.} & b^T y \\ \text{(PD)} & \text{S. a } A^T y + z = c \\ & z \geq 0 \\ & y \text{ irrestrito,} \end{array}$$

onde $y \in \mathbb{R}^m$ e $z \in \mathbb{R}^n$. Neste caso, já consideramos o problema dual no formato padrão pela adição da variável z (variável de folga).

O conjunto de soluções viáveis do problema (PD) é dado por:

$$D = \{(y, z) \in \mathbb{R}^m \times \mathbb{R}^n / A^T y + z = c, z \geq 0\}.$$

O interior de D definido por:

$$D^0 = \{(y, z) \in \mathbb{R}^m \times \mathbb{R}^n / A^T y + z = c, z > 0\},$$

é o conjunto das soluções viáveis interiores de (PD). Novamente, suporemos que $D^0 \neq \emptyset$.

A *teoria da dualidade* nos mostra a relação entre os problemas (PL) e (PD), ou seja, o conjunto de soluções primais nos fornece informações sobre o conjunto de soluções duais e vice-versa. Por exemplo, dados quaisquer vetores de soluções primais x para (PL) e (y, z) para (PD), temos:

$$b^T y \leq c^T x.$$

A função objetivo dual fornece um limite inferior para a função objetivo primal, e a função primal fornece um limite superior para a dual. As duas funções objetivo coincidem no valor, isto é: $b^T y^* = c^T x^*$, se e somente se x^* resolve (PL) e (y^*, z^*) resolve (PD) [Wright - 1997].

As condições de otimalidade tratadas neste trabalho são conhecidas como *Condições de Karush-Kunh-Tucker (ou KKT)*, e estão definidas no teorema 1 abaixo.

Teorema 1: (*Condições de Otimalidade de Karush-Kunh-Tucker - KKT*) [FP - 1993]:

Considere os problemas (PL) e (PD) acima. O vetor x^* é uma solução ótima do problema (PL) e (y^*, z^*) é solução ótima de (PD) se, e somente se:

$$A^T y + z = c, z \geq 0 \quad (\text{viabilidade dual}) \quad (2.1 \text{ a})$$

$$Ax = b, x \geq 0 \quad (\text{viabilidade primal}) \quad (2.1 \text{ b})$$

$$XZe = 0 \quad (\text{folgas complementares}) \quad (2.1 \text{ c})$$

onde $X = \text{diag}(x_1, x_2, \dots, x_n)$, $Z = \text{diag}(z_1, z_2, \dots, z_n)$ e $e = (1, 1, \dots, 1)^T$.

Do teorema 1, temos que o vetor (x^*, y^*, z^*) é solução ótima do sistema (2.1) se, e somente se x^* é solução ótima do (PL) e (y^*, z^*) é solução ótima do (PD). O vetor (x^*, y^*, z^*) é chamado de *solução primal-dual*.

O conjunto de soluções viáveis primais-duais e seu interior podem ser definidos como:

$$\Omega = \{ (x, y, z) / Ax = b, A^T y + z = c, (x, z) \geq 0 \}$$

$$\Omega^0 = \{ (x, y, z) / Ax = b, A^T y + z = c, (x, z) > 0 \},$$

respectivamente. Assumiremos $\Omega^0 \neq \emptyset$.

2.2 Método Primal-Dual

Os métodos primais-duais estão estritamente relacionados ao consagrado *Método de Newton* para soluções de sistemas de equações, lineares ou não. Uma iteração primal-dual é, em última instância, uma iteração adaptada do método de Newton, proveniente das condições de KKT (2.1).

Podemos escrever as condições de KKT na forma de uma equação, em conjunto com uma restrição. A equação provém das próprias condições de otimalidade e a restrição continua sendo a de não-negatividade. Obtemos então:

$$F(x, y, z) = \begin{bmatrix} Ax - b \\ A^T y + z - c \\ XZe \end{bmatrix} = 0, \quad (2.2 \text{ a})$$

$$(x, z) \geq 0, \quad (2.2 \text{ b})$$

com $F : \mathbb{R}^{2n+m} \rightarrow \mathbb{R}^{2n+m}$.

À equação (2.2 a) podemos aplicar o método de Newton para encontrar soluções viáveis (x^*, y^*, z^*) , porém respeitando o limite (2.2 b) *estritamente*, ou seja, $(x, z) > 0$, o que garante estarmos no interior da região viável (Ω^0) .

O método de Newton consiste em gerar uma seqüência de pontos $\{x^k\}$ que convirja para a solução de uma dada equação $G(x) = 0$, onde $G : \mathbb{R}^n \rightarrow \mathbb{R}^n, G \in C^1(\mathbb{R}^n)$ e $x \in \mathbb{R}^n$. O

método é baseado em uma expansão em primeira ordem (série de Taylor) da função considerada:

$$G(x) \approx G(x^k) + J(x^k)(x - x^k)$$

com J sendo o Jacobiano de G . Desta forma, se x^{k+1} é solução da expansão acima, então vale $J(x^k)s^k = -G(x^k)$, com $x^{k+1} = x^k + s^k$. Portanto, o método de Newton consiste em gerar os passos s^k e usá-los para atualizar o ponto x^k .

Uma iteração completa do método de Newton pode ser escrita como

$$x^{k+1} = x^k - J^{-1}(x^k)G(x^k).$$

Maiores referências sobre esse método podem ser encontradas em [MS – 1996].

Na aplicação do método de Newton surgem duas questões: (i) a necessidade de uma solução inicial e, (ii) a necessidade de exercermos algum controle sobre os passos, já que ele pode convergir para soluções de (2.2 a) mas que não satisfazem (2.2 b) estritamente (ditas soluções espúrias).

O item (i) será motivo de estudos posteriores (seção 2.5.1); por isso, consideraremos aqui que já temos uma solução inicial viável.

O item (ii) é o responsável pelos diversos algoritmos que trabalham com a estrutura primal-dual. Cada um deles possui uma maneira particular de limitar os passos de Newton, ou de alterar sua direção, mantendo a viabilidade da solução.

Escrevendo-se uma iteração de Newton para a equação (2.2 a), obtemos:

$$\begin{pmatrix} x^{k+1} \\ y^{k+1} \\ z^{k+1} \end{pmatrix} = \begin{pmatrix} x^k + \Delta x^k \\ y^k + \Delta y^k \\ z^k + \Delta z^k \end{pmatrix} = \begin{pmatrix} x^k \\ y^k \\ z^k \end{pmatrix} - J^{-1}(x^k, y^k, z^k) F(x^k, y^k, z^k), \quad (2.3 \text{ a})$$

ou ainda,

$$\begin{pmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{pmatrix} = -J^{-1}(x^k, y^k, z^k) F(x^k, y^k, z^k), \quad (2.3 \text{ b})$$



pré-multiplicando por $J(x^k, y^k, z^k)$:

$$J(x^k, y^k, z^k) \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = -F(x^k, y^k, z^k). \quad (2.3 \text{ c})$$

É imediato que:

$$J(x, y, z) = \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix}.$$

Portanto, para obter uma iteração de Newton para a equação (2.2 a), rescrevemos $J(x^k)s^k = -G(x^k)$ como:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = - \begin{bmatrix} Ax^k - b \\ A^T y^k + z^k - c \\ X_k Z_k e \end{bmatrix}.$$

Se o ponto corrente (x^k, y^k, z^k) é estritamente viável $(Ax^k = b, A^T y^k + z^k = c \text{ e } (x, z) > 0)$, então o sistema acima toma a forma:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -X_k Z_k e \end{bmatrix}.$$

Atualizando as variáveis (x, y, z) temos:

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k).$$

Os métodos primais-duais são variações do método de Newton obtidas a partir de modificações na maneira de calcular $(\Delta x^k, \Delta y^k, \Delta z^k)$, a direção de busca, e α_k , o parâmetro que controla o tamanho do passo em cada iteração. O valor de α_k pode atuar no algoritmo de dois modos:

- Mantendo o novo ponto no interior da região viável e,

- Cuidando para que (x, z) permaneça “longe” das fronteiras do quadrante não-negativo, $(x, z) \geq 0$. Direções calculadas a partir de pontos próximos a este quadrante tendem a ser distorcidas levando a pouco, se algum, progresso [Wright - 1997].

A idéia da *trajetória central* desempenha um papel fundamental no tratamento das questões acima.

2.2.1 A Trajetória Central

Uma das maneiras de manter o algoritmo do tipo primal-dual no interior da região viável é fazer com que a trajetória gerada pelo método de Newton esteja em alguma vizinhança da denominada *trajetória central* [Wright - 1997].

Matematicamente, a trajetória central C é um caminho de pontos estritamente viáveis, $(x_\tau, y_\tau, z_\tau) \in C \subset \Omega^0$, parametrizado por um escalar $\tau > 0$, de modo que:

$$A^T y + z = c \quad (\text{viabilidade dual}) \quad (2.4 \text{ a})$$

$$Ax = b \quad (\text{viabilidade primal}) \quad (2.4 \text{ b})$$

$$XZe = \tau e \quad (\text{folgas complementares}) \quad (2.4 \text{ c})$$

$$(x, z) > 0.$$

Fica claro que a única diferença entre (2.1) e (2.4) é a equação (2.4 c), onde trocamos a complementaridade $(x_i z_i = 0)$ pela exigência de que todos os produtos tenham o mesmo valor $(x_i z_i = \tau)$.

A trajetória central é um recurso interessante pois quando $\tau \rightarrow 0$, $(2.4) \rightarrow (2.1)$. Então, se C converge para algum ponto, deve ser para uma solução primal-dual do (PL). Desse modo, C atua como um guia, evitando soluções espúrias pois mantém os produtos $x_i z_i$ estritamente positivos, mas tendendo a zero à mesma velocidade.

Mais ainda: a maioria dos algoritmos primais-duais tomam direções em C com $\tau > 0$, ou seja, direções dentro do quadrante positivo definido por $(x, z) > 0$. Essa escolha de direções permite que passos maiores sejam dados em relação às direções de Newton puras, sem violar a condição de positividade, que chamaremos de *direção de busca*.



Para descrever a direção de busca modificada, introduziremos um *parâmetro de centralização* $\sigma \in [0,1]$ e uma *métrica dual* μ definida por:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i z_i = \frac{x^T z}{n}, \quad (2.5)$$

ou seja, uma média entre os produtos $x_i z_i$.

As equações de passo tomam agora a forma:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XZe + \sigma \mu e \end{bmatrix} \quad (2.6)$$

Através do controle de σ e μ podemos gerar uma vizinhança da trajetória central em que os novos passos calculados de acordo com (2.6) tenham melhor progresso do que os passos de Newton puros. Mas, por enquanto, apresentaremos apenas um teorema que garante a existência de um caminho central. Seu papel nos algoritmos ficará claro mais à frente.

Considere um conjunto estritamente viável H^0 :

$$H^0 = \{(x, z) / (x, y, z) \in \Omega^0 \text{ para algum } y \in \mathbb{R}^m\},$$

e uma função de barreira (com propriedades melhor definidas na seção 2.3) definida por:

$$f_\tau(x, z) = \frac{1}{\tau} x^T z - \sum_{i=1}^n \log(x_i z_i).$$

Teorema 2 [Wright - 1997]: Suponha $\Omega^0 \neq \emptyset$ e seja τ um número positivo. Então f_τ tem um único mínimo local em H^0 , que satisfaz as condições de trajetória central (2.4).

A prova do Teorema 2 encontra-se na referência citada.

Retornando às equações de passo (2.6), podemos escrevê-las em forma de sistema, como segue:



$$A^T \Delta y + \Delta z = 0 \quad (2.7 \text{ a})$$

$$A \Delta x = 0 \quad (2.7 \text{ b})$$

$$Z \Delta x + X \Delta z = -XZe + \sigma \mu e \quad (2.7 \text{ c})$$

Podemos isolar Δz de (2.7 c) :

$$\Delta z = -Ze + \sigma \mu X^{-1} e - X^{-1} Z \Delta x. \quad (2.7 \text{ d})$$

Substituindo (2.7d) em (2.7a), temos:

$$A^T \Delta y - X^{-1} Z \Delta x = Ze - \sigma \mu X^{-1} e \quad (2.7 \text{ e})$$

Escrevendo (2.7e) e (2.7b) na forma matricial, com Δz isolado temos:

$$\begin{bmatrix} A & 0 \\ -X^{-1}Z & A^T \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} 0 \\ -\sigma \mu X^{-1} e + Ze \end{bmatrix} \quad (2.8 \text{ a})$$

com

$$\Delta z = -Ze + \sigma \mu X^{-1} e - X^{-1} Z \Delta x. \quad (2.8 \text{ b})$$

Como X e Z são matrizes diagonais e não-singulares, podemos isolar Δx de (2.8), obtendo assim, um novo sistema equivalente:

$$AZ^{-1}XA^T \Delta y = A(Xe - \sigma \mu Z^{-1}e) \quad (2.9 \text{ a})$$

$$\Delta z = -A^T \Delta y \quad (2.9 \text{ b})$$

$$\Delta x = -Xe + \sigma \mu Z^{-1}e - Z^{-1}X \Delta z \quad (2.9 \text{ c})$$

O passo mais difícil é calcular (2.9 a), mas o primeiro termo envolve uma matriz definida positiva $\left((AD^2A^T) \right)$, onde $D = Z^{-1/2}X^{1/2}$, podendo então ser aplicada a Fatoração de Cholesky para resolver o sistema.

2.2.2 Uma Estrutura Primal-Dual

Através dos conceitos estudados até aqui, podemos definir uma estrutura geral para os algoritmos primais-duais. Os métodos abordados nas próximas seções são casos especiais desta estrutura.

Estrutura Primal-Dual [Wright - 1997]

Fase I

Passo 1: (*Inicialização*). Faça $k = 0$ e encontre $(x^0, y^0, z^0) \in \Omega^0$ uma solução inicial.

Fase II

Passo 2: (*Verificando otimalidade*). Se o ponto (x^k, y^k, z^k) satisfaz os critérios de otimalidade, **pare**.

Passo 3: (*Cálculo da direção de busca*). Seja $\sigma_k \in [0,1]$ e $\mu_k = \frac{(x^k)^T z^k}{n}$. Calcule:

$$\begin{aligned} AZ_k^{-1} X_k A^T \Delta y^k &= A(X_k e - \sigma_k \mu_k Z_k^{-1} e) \\ \Delta z^k &= -A^T \Delta y^k \\ \Delta x^k &= -X_k e + \sigma_k \mu_k Z_k^{-1} e - Z_k^{-1} X_k \Delta z^k \end{aligned} \quad (2.10)$$

Passo 4: (*Atualização*).

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k).$$

Faça $k = k + 1$ e volte ao passo 2.

Na Fase I do algoritmo calculamos uma solução inicial para o problema e, na Fase II, calculamos a direção de busca e atualizamos a solução até obtermos a solução ótima. No final deste capítulo abordaremos uma estrutura mais “completa” para o algoritmo de Ponto Interior Primal-Dual.

Na próxima subseção – seção 2.2.3 – faremos uma extensão do método Primal-Dual para Variáveis Canalizadas. Nas seções 2.3 e 2.4 apresentaremos as principais idéias dos métodos de *Redução de Potencial* e *Trajétória Central*, respectivamente. A análise detalhada destes métodos encontra-se nas referências citadas. E na seção 2.5 definiremos melhor os critérios para encontrar uma solução inicial e os critérios de parada do algoritmo.

2.2.3 Método de Pontos Interiores Primal-Dual para Variáveis Canalizadas

Nesta seção faremos uma extensão do método Primal-Dual para *variáveis canalizadas*, como segue.

Considere o problema primal:

$$\begin{array}{ll} \text{Min.} & c^T x \\ \text{(PL)} & \text{S. a } Ax = b \\ & 0 \leq x \leq u, \end{array}$$

onde $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $x, c, u \in \mathbb{R}^n$. Observe que não impomos uma restrição do tipo $l \leq x \leq u$ pois ela pode ser facilmente transformada em $0 \leq x \leq u$ (ver discussão ao final desta seção).

Transformando a inequação $x \leq u$, rescrevemos o problema (PL) como segue:

$$\begin{array}{ll} \text{Min.} & c^T x \\ \text{(PPL)} & \text{S. a } Ax = b \\ & x + s = u \\ & x \geq 0 \\ & s \geq 0. \end{array}$$

O problema dual de (PPL) é:

$$\begin{array}{ll} \text{Max.} & b^T y - u^T w \\ \text{(PPD)} & \text{S. a } A^T y + z - w = c \\ & z, w \geq 0 \\ & y \text{ irrestrito,} \end{array}$$

onde $y \in \mathbb{R}^m$ e $z, w \in \mathbb{R}^n$.

As condições de otimalidade de KKT para os problemas (PPL) e (PPD) são:

$$A^T y + z - w = c, \quad (2.11 \text{ a})$$

$$Ax = b, \quad (2.11 \text{ b})$$

$$x + s = u, \quad (2.11 \text{ c})$$

$$XZe = 0, \quad (2.11 \text{ d})$$

$$SWe = 0, \quad (2.11 \text{ e})$$

onde X, Z, S e W são matrizes diagonais, $e = (1, \dots, 1)^T$ e μ a métrica dual (2.5).

O conjunto solução, Ω^0 , para os problemas (PPL) e (PPD) será denotado aqui como:

$$\Omega^0 = \{(x, y, s, w, z) / A^T y + z - w = c, Ax = b, x + s = u, (x, s, w, z) > 0\}.$$

Supondo que as condições de otimalidade (2.11) satisfaçam a viabilidade primal e dual, aplicamos os passos de Newton para encontrar a direção de busca. Então, escrevendo (2.11) em forma de função temos:

$$F(x, y, s, w, z) = \begin{bmatrix} Ax - b \\ A^T y + z - w - c \\ x + s - u \\ SWe \\ XZe \end{bmatrix}.$$

Denotaremos por: $r_b = Ax - b$, $r_c = A^T y + z - w - c$ e $r_u = x + s - u$.

Respeitando os mesmos passos usados em (2.3), obtemos o sistema para os passos de Newton, como segue:

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ 0 & A^T & 0 & -I & I \\ I & 0 & I & 0 & 0 \\ 0 & 0 & We & Se & 0 \\ Ze & 0 & 0 & 0 & Xe \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta s^k \\ \Delta w^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c \\ -r_u \\ -SWe + \mu e \\ -XZe + \mu e \end{bmatrix}. \quad (2.12)$$

A forma acima é pouco conveniente devido à estrutura esparsa e de blocos da matriz. Portanto, resolvemos para cada Δx , Δs , Δy , Δw e Δz separadamente:

$$A\Delta x^k = -r_b, \quad (2.12 \text{ a})$$

$$A^T \Delta y^k + \Delta z^k - \Delta w^k = -r_c, \quad (2.12 \text{ b})$$

$$\Delta x^k + \Delta s^k = -r_u, \quad (2.12 \text{ c})$$

$$We\Delta s^k + Se\Delta w^k = -SWe + \mu e, \quad (2.12 \text{ d})$$

$$Ze\Delta x^k + Xe\Delta z^k = -ZXe + \mu e. \quad (2.12 \text{ e})$$

Tomando a i -ésima componente do vetor (2.12 c), temos:

$$\Delta x_i^k + \Delta s_i^k = -r_u^i.$$

Para simplificar a notação, denotaremos por:

$$\Delta x = \Delta x_i^k, \Delta s = \Delta s_i^k \text{ e } u = -r_u^i.$$

Tomando $x + s = u$, podemos escrever x como função de s : $x(s) = u - s$. Se calcularmos:

$$\Delta x = x(s + \Delta s) - x(s) = u - (s + \Delta s) - u + s = -\Delta s.$$

Portanto: $\Delta x = -\Delta s$, ou ainda:

$$\Delta s^k = -\Delta x^k. \quad (2.13 \text{ a})$$

Agora, vamos isolar Δw e Δz de (2.12 d) e (2.12 e), respectivamente.

$$We\Delta s^k + Se\Delta w^k = -SWe + \mu e$$

$$\Delta w^k = S^{-1}(-SWe + \mu e - We\Delta s^k), \quad (2.13 \text{ b})$$

$$Ze\Delta x^k + Xe\Delta z^k = -ZXe + \mu e$$

$$\Delta z^k = X^{-1}(-XZe + \mu e - Ze\Delta x^k). \quad (2.13 \text{ c})$$

Substituindo (2.13 a) – (2.13 c) em (2.12 b) temos Δy e Δx :

$$A^T \Delta y^k + \Delta z^k - \Delta w^k = -r_c$$

$$(A\theta A^T) \Delta y^k = -r_b + A\theta(\rho(\mu) - r_c), \quad (2.13 \text{ d})$$

$$A^T \Delta y^k + \Delta z^k - \Delta w^k = -r_c$$

$$\Delta x^k = \theta(A^T \Delta y^k - \rho(\mu) - r_c), \quad (2.13 \text{ e})$$

onde:

$$\theta = (X^{-1}Z + S^{-1}W)^{-1},$$

e

$$\rho(\mu) = \mu(S^{-1} - X^{-1})e - (W - Z)e.$$

Para pontos em $(x, y, z) \in \Omega^0$, $r_b = 0$, $r_c = 0$ e $r_u = 0$.

Resumindo (2.13), temos:

$$\begin{aligned} \Delta s^k &= -\Delta x^k, \\ \Delta w^k &= S^{-1}(-SWe + \mu e - We\Delta s^k), \\ \Delta z^k &= X^{-1}(-XZe + \mu e - Ze\Delta x^k), \\ (A\theta A^T)\Delta y^k &= -r_b + A\theta(\rho(\mu) - r_c), \\ \Delta x^k &= \theta(A^T \Delta y^k - \rho(\mu) - r_c). \end{aligned} \quad (2.13)$$

Após a discussão acima podemos elaborar o algoritmo, que é dado abaixo.

Algoritmo de Pontos Interiores Primal-Dual para Variáveis Canalizadas [BM -1992]

Fase I

Passo 1: (*Inicialização*). Faça $k = 0$ e $(x^0, y^0, s^0, w^0, z^0) \in \Omega^0$.

Fase II

Passo 2: (*Verificação da Otimalidade*). Se $(x^k, y^k, s^k, w^k, z^k)$ satisfaz os critérios de otimalidade, **pare**.

Passo 3: (*Cálculo de μ*). Calcule $\mu_k = \frac{1}{n} \sum_{i=1}^n x_i^k z_i^k$.

Passo 4: Calcule: $\theta = (X^{-1}Z + S^{-1}W)^{-1}$ e $\rho(\mu) = \mu(S^{-1} - X^{-1})e - (W - Z)e$.

Passo 5: (*Cálculo dos passos do método de Newton*). Calcule Δx , Δy , Δs , Δw e Δz usando (2.13), ou seja:

$$\begin{aligned}\Delta s^k &= -\Delta x^k, \\ \Delta w^k &= S_k^{-1}(-SWe + \mu e - We\Delta s^k), \\ \Delta z^k &= X_k^{-1}(-XZe + \mu e - Ze\Delta x^k), \\ (A\theta A^T)\Delta y^k &= -r_b + A\theta(\rho(\mu) - r_c), \\ \Delta x^k &= \theta(A^T\Delta y^k - \rho(\mu) - r_c).\end{aligned}$$

Passo 6: (*Cálculo do tamanho do passo*). Encontre α_k tal que: $x^k + \alpha_k \Delta x^k > 0$, $s^k + \alpha_k \Delta s^k > 0$, $w^k + \alpha_k \Delta w^k > 0$ e $z^k + \alpha_k \Delta z^k > 0$.

Passo 7: (*Atualização*). Seja:

$$(x^{k+1}, s^{k+1}, y^{k+1}, z^{k+1}, w^{k+1}) = (x^k, s^k, y^k, z^k, w^k) + \alpha_k (\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k).$$

Faça $k = k + 1$ e volte ao passo 2.

Variáveis Limitadas Inferiormente

No começo desta seção foi citado que uma restrição do tipo $l \leq x \leq u$ pode ser facilmente transformada em $0 \leq x \leq u$. Discutiremos aqui o porquê desta afirmação.

A restrição $l \leq x \leq u$ pode ser “colocada” no problema da seguinte maneira:

$$x \leq u \text{ e } x \geq l.$$

A segunda restrição é pouco conveniente devido a dificuldade computacional, então fazemos uma transformação, e quando encontramos a solução ótima aplicamos a transformação inversa para os valores originais.

Seja $Ax = b$ a matriz de restrições e $l \leq x \leq u$ a restrição adicional. Então:

$$l \leq x \leq u \Rightarrow l - l \leq x - l \leq u - l \Rightarrow 0 \leq x - l \leq u - l.$$

Tomando: $\bar{u} = u - l$ e $\bar{x} = x - l$ ou $x = \bar{x} + l$, temos: $0 \leq \bar{x} \leq \bar{u}$.

Substituindo no problema original, teremos:

$$Ax = b, 0 \leq \bar{x} \leq \bar{u} \Rightarrow A(\bar{x} + l) = b \Rightarrow A\bar{x} = b - Al.$$

Observe que fazer este tipo de transformação tem um custo de n^2 operações.

2.3 Método de Redução de Potencial (RP)

O *Método de Redução de Potencial* é uma variante dos algoritmos que usam passos de Newton. Uma característica deste método é que os pontos evitam seguir a trajetória central, ou seja, caminham independentemente da trajetória central discutida na seção 2.2.1.

Este método é parte de uma classe maior de métodos denominados *Métodos de Penalização*. De um modo bem geral, um método de penalização modifica a função objetivo acrescentando um termo que (i) aumente muito seu valor quando determinada condição de controle é violada e (ii) tende a zero quando nos aproximamos da solução ótima. Este termo é denominado usualmente de *função de barreira* ou *função potencial* [MS – 1996].

A vantagem do uso da função potencial em métodos de Pontos Interiores é que podemos exercer um controle maior das iterações que geram a sequência de pontos interiores, “estreitando” o caminho por onde a trajetória gerada pode passar. Por isso ela tem desempenhado um papel importante no desenvolvimento de algoritmos de Pontos Interiores. Karmarkar usou uma função potencial para a Programação Linear sob a forma:

$$\rho \log(c^T x - L) - \sum_{i=1}^n \log x_i,$$

onde $\rho = n + 1$ e L é um limite inferior para o valor da função objetivo, provando sua convergência e resultados de complexidade por mostrar que sua função diminui, pelo menos um valor constante em cada iteração. Por exemplo, para o problema:

$$\begin{array}{ll} \text{Min.} & x_1 + x_2 \\ \text{S. a} & x_1 + x_3 = 1 \\ & x_2 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

cujo politopo é um quadrado de lado 1 no plano (x_1, x_2) , a função de Karmarkar torna-se: $\Phi(x_1, x_2) = 3 \log(x_1 + x_2) - \log x_1 - \log x_2$. Observe na figura 2.1 abaixo como a função decresce rapidamente na imediações do ótimo, $x^* = (0,0)$.

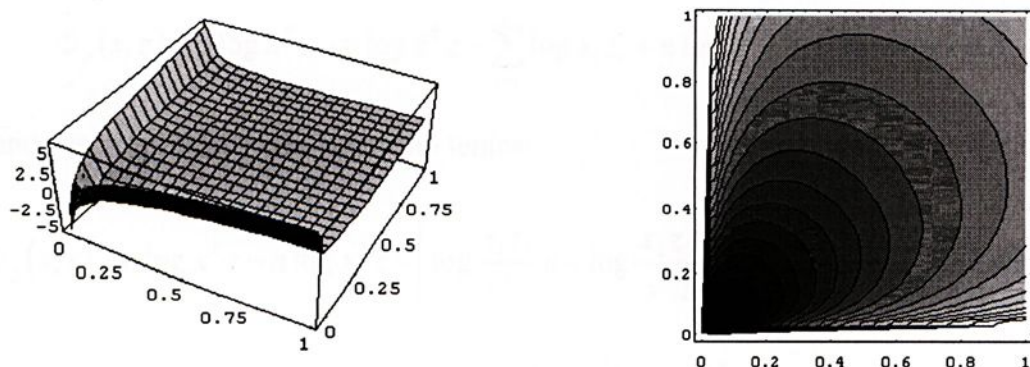


Figura 2.1 – A função potencial e suas curvas de nível.

Depois da publicação do algoritmo de Karmarkar, houve muitos avanços nos algoritmos que usam esta função; porém, a maioria das novas técnicas ainda baseava-se nos valores das variáveis primais. A partir de 1987 os algoritmos passaram a utilizar de forma mais balanceada variáveis primais e duais. Nessa linha temos a função de potencial primal-dual de *Tanabe-Todd-Ye*, definida por:

$$\Phi_{\rho}(x, z) = \rho \log x^T z - \sum_{i=1}^n \log x_i z_i, \quad (2.14)$$

para um parâmetro $\rho > n$. O trabalho destes pesquisadores foi crucial no desenvolvimento posterior dos métodos de RP [Wright - 1997].

Os algoritmos baseados em Φ_{ρ} usam a função potencial apenas para monitorar o “progresso” e provar a convergência [Wright - 1997]. Este progresso refere-se ao potencial atribuído a cada ponto de Ω^0 por Φ_{ρ} e que diminui a medida que nos aproximamos da solução. Como todos os algoritmos baseados na estrutura primal-dual, discutida na seção anterior, o algoritmo de redução de potencial obtém sua direção de busca pela execução de (2.10) para algum $\sigma_k \in (0, 1)$. O tamanho do passo α_k é escolhido para minimizar Φ_{ρ} ao longo da direção calculada.

Voltando à função (2.14), vamos fazer algumas manipulações algébricas para extrairmos algumas propriedades de Φ_{ρ} . Primeiramente vamos somar e subtrair dois novos termos, como segue:

$$\Phi_\rho(x, z) = \rho \log x^T z - n \log x^T z - \sum_{i=1}^n \log x_i z_i + n \log x^T z - n \log n + n \log n.$$

Aplicando as propriedades de logaritmo temos:

$$\begin{aligned}\Phi_\rho(x, z) &= \rho \log x^T z - n \log x^T z - \left[\log \frac{x_1 z_1}{x^T z} n + \log \frac{x_2 z_2}{x^T z} n + \dots + \log \frac{x_n z_n}{x^T z} n \right] + n \log n, \\ \Rightarrow \Phi_\rho(x, z) &= (\rho - n) \log x^T z - \sum_{i=1}^n \log \frac{x_i z_i}{x^T z / n} + n \log n.\end{aligned}$$

Ou seja, podemos rescrever a função (2.14) da seguinte maneira:

$$\Phi_\rho(x, z) = (\rho - n) \log x^T z - \sum_{i=1}^n \log \frac{x_i z_i}{x^T z / n} + n \log n \quad (2.15)$$

De (2.15) percebemos que Φ_ρ faz o balanceamento entre dualidade e centralidade. Se $x^T z$ é pequeno, favorecemos a dualidade e se nenhum $x_i z_i$ é muito menor do que a média $\mu = \frac{x^T z}{n}$, favorecemos a centralidade.

Duas propriedades importantes podem ser então observadas:

1. $\Phi_\rho \rightarrow +\infty$ se $x_i z_i \rightarrow 0$, para algum i , mas $\mu = \frac{x^T z}{n}$ não tende a zero;
2. $\Phi_\rho \rightarrow -\infty$ se, e somente se $(x, y, z) \rightarrow \Omega^0$.

De acordo com as condições de KKT, só estaremos no ótimo se $x^T z = 0$. Portanto, é desejável que o método aproveite a função de barreira de modo a forçar $\mu \rightarrow 0$.

A propriedade (1) é o efeito de barreira, pois para que $\mu \rightarrow 0$, **todas** as componentes de $x^T z$ devem ir a zero. Caso só algumas tendam a zero, $\Phi_\rho \rightarrow +\infty$, funcionando então como uma barreira.

Por outro lado, o termo $\left(-\sum_{i=1}^n \log \frac{x_i z_i}{x^T z / n} + n \log n \right)$ tem um limite inferior [Wright - 1997]. Portanto, $\Phi_\rho \rightarrow -\infty$ se, e somente se $(\rho - n) \log x^T z \rightarrow -\infty$, o que implica que

$x^T z \rightarrow 0$. Então, o método de Redução de Potencial gera iterações no sentido de levar $\mu \rightarrow 0$, usando informações fornecidas por estas duas propriedades.

Algoritmo de Redução de Potencial Primal-Dual

Este algoritmo toma $\sigma_k = \frac{n}{\rho}$, onde $\rho > n$ é o parâmetro de (2.15). O passo α_k é escolhido para minimizar $\Phi_\rho(\cdot)$ ao longo da direção de busca e mantendo a viabilidade.

Algoritmo RP [Wright - 1997]

Fase I

Passo 1: (*Inicialização*). Faça $k = 0$, $\rho > n$ e encontre $(x^0, y^0, z^0) \in \Omega^0$;

Fase II

Passo 2: (*Verificando otimalidade*). Se o ponto (x^k, y^k, z^k) satisfaz os critérios de otimalidade, **pare**.

Passo 3: (*Cálculo da direção de busca*). Seja $\sigma_k = \frac{n}{\rho}$ e resolva (2.10) para obter

$$(\Delta x^k, \Delta y^k, \Delta z^k);$$

Passo 4: (*Cálculo do tamanho do passo*). Calcule:

$$\alpha_{\max} = \sup\{\alpha \in [0, 1] / (x, z) + \alpha(\Delta x, \Delta z) \geq 0\},$$

e escolha α_k de $\alpha_k = \arg \min_{\alpha \in (0, \alpha_{\max})} \Phi_\rho(x^k + \alpha \Delta x^k, z^k + \alpha \Delta z^k)$;

Passo 5: (*Atualização*).

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k);$$

Faça $k = k + 1$ e volte ao passo 2.

Como foi citado anteriormente, este algoritmo se diferencia da estrutura Primal-Dual, da seção 2.2.2, pela mudança no valor de σ_k e pela escolha do tamanho do passo α_k , como pode ser observado acima.

2.4 Método da Trajetória Central

Os algoritmos do tipo *Trajetória Central* (ou *Path-Following*, como são conhecidos), restringem as suas iterações a uma vizinhança da trajetória central $\mathbf{C}$ (seção 2.2.1) [Gonzaga – 1989]. A vizinhança da Trajetória Central elimina pontos (x, z) que estão próximos da fronteira da região viável e calcula a direção de busca de qualquer ponto da vizinhança fazendo, pelo menos, progresso na direção do conjunto solução.

Nestes algoritmos, usaremos a métrica dual $\mu = x^T z / n$, como definida em (2.5), para definir o quão desejável é um ponto no espaço de busca. Esta métrica dual μ_k (calculada a cada iteração) é forçada a zero quando $k \rightarrow \infty$, e assim o ponto (x^k, y^k, z^k) satisfaz as condições de KKT definidas em (2.1).

As duas vizinhanças mais importantes de $\mathbf{C}$ são:

1. a vizinhança de norma-2, $\aleph_2(\theta)$, definida por:

$$\aleph_2(\theta) = \{(x, y, z) \in \Omega^0 / \|XZe - \mu e\| \leq \theta \mu\} \quad (2.16)$$

para algum $\theta \in (0, 1)$;

2. e a vizinhança de norma- ∞ , $\aleph_{-\infty}(\theta)$, definida por:

$$\aleph_{-\infty}(\theta) = \{(x, y, z) \in \Omega^0 / x_i z_i \geq \theta \mu \text{ para todo } i = 1, 2, \dots, n\} \quad (2.17)$$

para algum $\theta \in (0, 1)$.

Observe que na vizinhança $\aleph_{-\infty}(\theta)$ os parâmetros $x_i z_i \geq \theta \mu$ são escolhidos de maneira que não se aproximam da fronteira da região viável.

A cor cinza na figura 2.2 abaixo ilustra duas possíveis regiões:

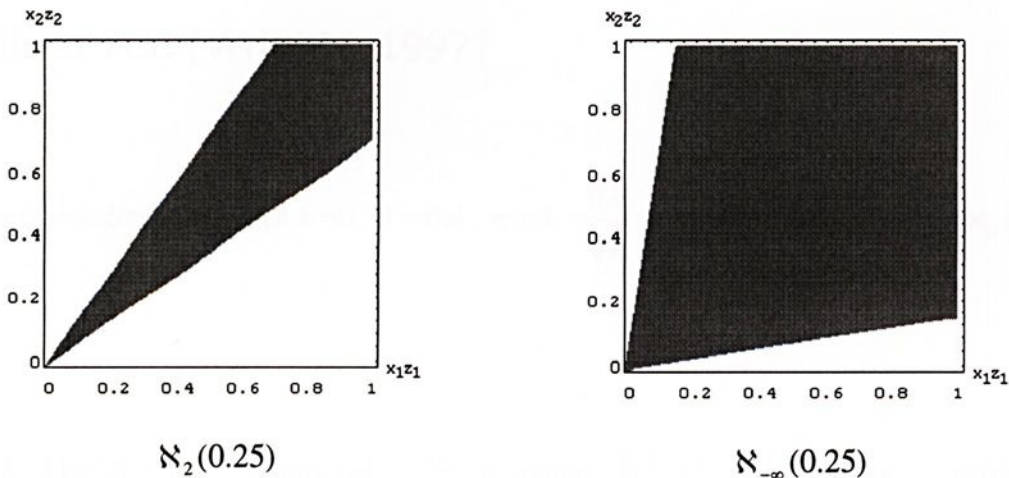


Figura 2.2 – Exemplo de vizinhanças da Trajetória Central

Os métodos da Trajetória Central seguem a Estrutura Primal-Dual definida na seção 2.2.2. Estes métodos selecionam uma das vizinhanças definidas em (2.16) e (2.17) e escolhem o parâmetro de centralização σ e o tamanho do passo α , para garantir que em cada iteração o ponto (x^k, y^k, z^k) esteja dentro da vizinhança escolhida [Wright - 1997].

Nesta seção descreveremos dois métodos da Trajetória Central que utilizam a vizinhança $\aleph_2(\theta)$: o *Método da Trajetória Central de Passo Curto*, e o *Método Preditor Corretor*. Abordaremos também um método que usa a vizinhança $\aleph_{-\infty}(\theta)$, o *Método da Trajetória Central de Passo Longo*, e uma variante deste último método, o *Método de Ponto Interior Inviável*.

2.4.1 Método da Trajetória Central de Passo Curto (TCC)

Este método se caracteriza pela sua simplicidade: ele inicia no ponto $(x^0, y^0, z^0) \in \aleph_2(\theta)$ e usa valores uniformes $\alpha_k = 1$ e $\sigma_k = \sigma$, onde θ e σ satisfazem a uma certa relação, descrita no algoritmo abaixo. Em toda iteração o ponto (x^k, y^k, z^k) situa-se em $\aleph_2(\theta)$, e a métrica dual μ_k converge linearmente para zero com razão constante $1 - \sigma$.

Algoritmo TCC [Wright - 1997]

Fase I

Passo 1: (*Inicialização*). Faça $k=0$, $\theta=0.4$, $\sigma=1-\frac{0.4}{\sqrt{n}}$ e encontre $(x^0, y^0, z^0) \in \aleph_2(\theta)$.

Fase II

Passo 2: (*Verificando otimalidade*). Se o ponto (x^k, y^k, z^k) satisfaz os critérios de otimalidade, **pare**.

Passo 3: (*Cálculo da direção de busca*). Seja $\sigma_k = \sigma$, resolva (2.10) para obter $(\Delta x^k, \Delta y^k, \Delta z^k)$;

Passo 4: (*Atualização*).

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + (\Delta x^k, \Delta y^k, \Delta z^k);$$

Volte ao passo 2.

De acordo com [Wright – 1997] os parâmetros θ e σ são escolhidos de modo a garantir a convergência linear e global do método.

2.4.2 Método Preditor – Corretor (PC)

No algoritmo PC escolhemos $\sigma_k \equiv \sigma$ estritamente 0 ou 1. Esta escolha tem a dupla finalidade de melhorar a centralização e reduzir a métrica dual μ em um único passo. O algoritmo PC toma dois tipos diferentes de passos para alcançar cada um desses objetivos, que são:

- Passo *preditor* ($\sigma_k = 0$), para reduzir μ ;
- Passo *corretor* ($\sigma_k = 1$), para melhorar a centralização.

Uma outra característica deste algoritmo é o par de vizinhanças $\aleph_2$, aninhadas uma dentro da outra. As iterações de índices pares $\{(x^k, y^k, z^k), \text{com } k \text{ par}\}$ ficam no interior da vizinhança, ao passo que as iterações de índices ímpares desviam-se para a vizinhança

externa. Podemos definir a vizinhança interna como $\aleph_2(0.25)$ e a externa como $\aleph_2(0.5)$. Outras escolhas podem ser feitas, se garantirmos que as duas vizinhanças estejam relacionadas uma com a outra como no algoritmo abaixo [Wright - 1997].

Algoritmo PC [Wright - 1997]

Fase I

Passo 1: (*Inicialização*). Faça $k = 0$ e escolha $(x^0, y^0, z^0) \in \aleph_2(0.25)$.

Fase II

Passo 2: (*Verificando otimalidade*). Se o ponto (x^k, y^k, z^k) satisfaz os critérios de otimalidade, **pare**.

Passo 3: (*Passo preditor*). Se k é par:

Resolva (2.10) com $(\sigma_k = 0)$ para obter $(\Delta x^k, \Delta y^k, \Delta z^k)$;

Escolha α_k como o maior valor de α em $[0, 1]$, tal que

$$(x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k) \in \aleph_2(0.5);$$

Vá ao passo 5.

Passo 4: (*Passo corretor*). Se k é ímpar:

Resolva (2.10) com $(\sigma_k = 1)$ para obter $(\Delta x^k, \Delta y^k, \Delta z^k)$;

Vá ao passo 5.

Passo 5: (*Atualização*).

Se k é par:

$$\text{Faça } (x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k);$$

Faça $k = k + 1$ e volte ao passo 2.

Se k é ímpar:

$$\text{Faça } (x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + (\Delta x^k, \Delta y^k, \Delta z^k);$$

Faça $k = k + 1$ e volte ao passo 2.

2.4.3 Método da Trajetória Central de Passo Longo (TCL)

O algoritmo da Trajetória Central de Passo Longo gera uma sequência de pontos na vizinhança $\aleph_{-\infty}(\theta)$, com θ pequeno (um valor típico para este parâmetro é $\theta = 10^{-3}$), reduzindo os seus elementos estritamente ao conjunto viável Ω^0 .

Em cada iteração, escolhemos um parâmetro de centralização σ_k que situa-se entre dois valores fixos $\sigma_{\min}$ e $\sigma_{\max}$, onde $0 < \sigma_{\min} < \sigma_{\max} < 1$. A direção de busca é calculada por (2.10), como nos outros métodos, e escolhemos o tamanho do passo α_k como o maior possível, localizando-se dentro de $\aleph_{-\infty}(\theta)$.

Algoritmo (TCL) [Wright - 1997]

Fase I

Passo 1: (*Inicialização*). Faça $k = 0$, escolha θ , $\sigma_{\min}$, $\sigma_{\max}$ com $\theta \in (0, 1)$, $0 < \sigma_{\min} < \sigma_{\max} < 1$, e

$$(x^0, y^0, z^0) \in \aleph_{-\infty}(\theta).$$

Fase II

Passo 2: (*Verificando otimalidade*). Se o ponto (x^k, y^k, z^k) satisfaz os critérios de otimalidade, **pare**.

Passo 3: (*Cálculo da direção de busca*). Escolha $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$;

Resolva (2.10) para obter $(\Delta x^k, \Delta y^k, \Delta z^k)$;

Passo 4: (*Atualização*). Escolha α_k como o maior valor de α em $[0, 1]$ tal que:

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k) \in \aleph_{-\infty}(\theta);$$



$$\text{Faça } (x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k).$$

Faça $k = k + 1$ e volte ao passo 2.

Os métodos discutidos nesta seção são variantes da estrutura Primal-Dual da seção 2.2.2, e seguem explicitamente a trajetória central C. Cada um dos algoritmos desta seção se diferencia por usar as vizinhanças $\mathcal{N}_{-\infty}(\theta)$ e $\mathcal{N}_2(\theta)$ de maneira distinta.

2.4.4 Método de Pontos Interiores Inviável (PII)

Para todos os métodos discutidos até aqui, supomos que temos um ponto inicial $(x^0, y^0, z^0) \in \Omega^0$. Contudo, para muitos problemas, um ponto inicial é muito difícil de ser obtido. Podemos obtê-lo pela reformulação do problema, porém esta estratégia pode trazer distorções, tornando-o de difícil resolução.

Na próxima seção (seção 2.5) veremos o *Método M-Grande* que é usado para obter um ponto inicial. Aqui trataremos do *Método de Pontos Interiores Inviável*, o qual requer apenas que as variáveis x^0 e z^0 sejam estritamente positivas. A direção de busca sofre uma modificação para mover-se em direção à região viável e obter uma centralização. Porém, é necessária uma única mudança nas equações de (2.6). Definiremos os resíduos para as duas equações lineares:

$$r_b = Ax - b \text{ e } r_c = A^T y + z - c.$$

Assim, (2.6) tem a forma:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = \begin{bmatrix} -r_b \\ -r_c \\ -X_k Z_k e + \sigma \mu e \end{bmatrix}.$$

Seguindo os passos (2.7) e (2.9), obtemos as seguintes equações:

$$AZ^{-1}XA^T \Delta y = -r_b + A(x - Z^{-1}Xr_c - \sigma \mu Z^{-1}e), \quad (2.18 \text{ a})$$

$$\Delta z = -r_c - A^T y \quad (2.18 \text{ b})$$

$$\Delta x = -x + \sigma \mu Z^{-1}e - Z^{-1}X\Delta z \quad (2.18 \text{ c})$$

A direção de busca (2.18) é ainda um passo de Newton em direção ao ponto $(x_\tau, y_\tau, z_\tau) \in \mathcal{C}$. O método tenta eliminar a inviabilidade em um único passo. Se tomarmos $\alpha = 1$, os resíduos r_b e r_c tornam-se zero, e todas as demais iterações permanecem estritamente viáveis. Porém devemos respeitar duas condições importantes para a escolha de α : (i) a quantidade na qual as restrições $Ax = b$ e $A^T y + z = c$ são violadas pelos resíduos precisam ser diminuídas tão rápido quanto μ , e (ii) fazer μ decrescer uma pequena quantia prevista em cada iteração, usando, por exemplo, a condição do tipo Armijo ($\mu_{k+1} \leq (1 - 0.1\alpha)\mu_k$). Ou seja, a condição de Armijo é imposta para garantir o decréscimo de μ , outros critérios podem ser usados, veja [Wright - 1997].

O método de Pontos Interiores Inviável trabalha com uma vizinhança da trajetória central $\aleph_{-\infty}(\theta, \beta)$, que é uma extensão de $\aleph_{-\infty}(\theta)$, definida em (2.17), a qual inclui pontos inviáveis. Esta vizinhança é definida por:

$$\aleph_{-\infty}(\theta, \beta) = \left\{ (x, y, z) / \|r_b, r_c\| \leq \left[\frac{\|r_b^0, r_c^0\|}{\mu_0} \right] \beta \mu, (x, z) > 0, x_i z_i \geq \theta \mu, i = 1, 2, \dots, n \right\}, \quad (2.19)$$

onde $\theta \in (0, 1)$ e $\beta \geq 1$ são parâmetros dados, (r_b^0, r_c^0) e μ_0 são calculados no ponto inicial (x^0, y^0, z^0) . Note que:

$$\beta \geq 1 \quad (2.20)$$

garante que o ponto inicial $(x^0, y^0, z^0) \in \aleph_{-\infty}(\theta, \beta)$.

Para todos os pontos na vizinhança $\aleph_{-\infty}(\theta, \beta)$, a inviabilidade é uniformemente limitada por algum múltiplo da métrica dual μ . Forçando μ_k tender a zero monotonicamente e restringindo todas as iterações (x^k, y^k, z^k) à vizinhança $\aleph_{-\infty}(\theta, \beta)$, o algoritmo de Pontos Interiores Inviável garante que $r_b^k \rightarrow 0$ e $r_c^k \rightarrow 0$, quando $k \rightarrow \infty$. A outra condição de (2.19), $x_i z_i \geq \theta \mu$, tem a mesma finalidade que na vizinhança $\aleph_{-\infty}(\theta)$. Esta condição mantém o balanço entre os pares $x_i z_i$ evitando direções com componentes que aproximam-se de zero prematuramente, o que distorceria as direções de Newton.

Um algoritmo para o método de Pontos Interiores Inviável pode ser formulado como segue.



Algoritmo (PII) [Wright - 1997]

Fase I

Passo 1: (*Inicialização*). Faça $k = 0$. Escolha $\theta, \beta, \sigma_{\min}, \sigma_{\max}$ tal que $\theta \in (0, 1), \beta \geq 1, 0 < \sigma_{\min} < \sigma_{\max} < 0.5$; escolha (x^0, y^0, z^0) tal que $(x^0, z^0) > 0$.

Fase II

Passo 2: (*Verificando otimalidade*). Se o ponto (x^k, y^k, z^k) satisfaz os critérios de otimalidade, **pare**.

Passo 3: (*Cálculo da direção de busca*). Escolha $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$;

Resolva (2.18) para obter $(\Delta x^k, \Delta y^k, \Delta z^k)$;

Passo 4: (*Atualização*). Escolha α_k como o maior valor de α em $[0, 1]$ tal que:

$$(x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k) \in \mathcal{N}_{-\infty}(\theta, \beta);$$

e a seguinte condição de Armijo seja satisfeita:

$$\mu_{k+1} \leq (1 - 0.1\alpha)\mu_k$$

Faça $(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k)$.

Faça $k = k + 1$ e volte ao passo 2.

2.5 Comentários Práticos sobre o Algoritmo de Pontos Interiores Primal-Dual

Na seção 2.2.2 definimos uma estrutura geral para o algoritmo Primal-Dual, e nas seções 2.3 e 2.4 definimos algumas variantes desta estrutura. Porém, durante estas discussões, tomamos como conhecida uma solução inicial e omitimos o critério de parada do algoritmo. Trataremos nesta seção destes tópicos, bem como de uma estrutura mais “completa” para o algoritmo Primal-Dual.

2.5.1 Solução Inicial

Até aqui, discutimos sobre algoritmos de Pontos Interiores, supondo conhecida uma solução inicial. A dificuldade em obter uma solução inicial é equivalente a de aplicar o algoritmo até a solução ótima; por isso, muitos algoritmos teóricos assumem a existência de um ponto inicial e o usam para inicializar o algoritmo. O processo de encontrar uma solução inicial é conhecido como “Fase I” do algoritmo, e o processo de resolver o algoritmo até a otimalidade, de “Fase II” [Levkovitz – 1992]. Então, nesta seção, abordaremos a Fase I do algoritmo Primal-Dual.

Seja (x^0, y^0, z^0) com $(x^0, z^0) > 0$ uma solução primal-dual inicial inviável. Para obter uma solução viável considere os seguintes problemas primal e dual artificiais:

$$\begin{aligned}
 & \text{Min. } c^T x + M_P x_{n+1} \\
 & \text{S. a } Ax + (b - Ax^0)x_{n+1} = b \\
 & \quad (A^T y^0 + z^0 - c)^T x + x_{n+2} = M_D \\
 & \quad (x, x_{n+1}, x_{n+2}) \geq 0,
 \end{aligned}
 \tag{PLA}$$

e

$$\begin{aligned}
 & \text{Max. } b^T y + M_D y_{m+1} \\
 & \text{S. a } A^T y + (A^T y^0 + z^0 - c)y_{m+1} + z = c \\
 & \quad (b - Ax^0)^T y + z_{n+1} = M_P \\
 & \quad y_{m+1} + z_{n+2} = 0 \\
 & \quad (z, z_{n+1}, z_{n+2}) \geq 0,
 \end{aligned}
 \tag{PDA}$$

onde x_{n+1} e y_{m+1} são variáveis artificiais dos problemas primal e dual, respectivamente, com seus coeficientes de custo artificiais M_P e M_D . O coeficiente dessas variáveis na primeira restrição de (PLA) e na primeira de (PDA) correspondem a uma medida de inviabilidade da aproximação inicial. A variável x_{n+2} corresponde à variável dual artificial y_{m+1} , e z_{n+1}, z_{n+2} são as variáveis duais de folga correspondentes a x_{n+1} e x_{n+2} . Essas variáveis são necessárias para manter a relação de dualidade entre os dois problemas.

Fixando-se M_p e M_D com valores que dominam a função objetivo primal e dual, garantimos que a direção de busca é dominada pela direção viável. Se um ponto viável existe, então durante o processo iterativo as variáveis artificiais gradualmente convergem para zero [Levkovitz – 1992].

Podemos escolher M_p e M_D de maneira que:

$$M_p > (b - Ax^0)^T y^0, \quad (2.21 \text{ a})$$

e

$$M_D > (A^T y + z^0 - c)^T x^0, \quad (2.21 \text{ b})$$

então $(x^0, x_{n+1}^0, x_{n+2}^0)$ e $(y^0, y_{m+1}^0, z^0, z_{n+1}^0, z_{n+2}^0)$ são soluções viáveis para os problemas artificiais (PLA) e (PDA), respectivamente.

Os valores de $x_{n+1}^0, x_{n+2}^0, y_{m+1}^0, z_{n+1}^0$ e z_{n+2}^0 podem ser obtidos como segue.

Seja $Ax + (b - Ax^0)x_{n+1}^0 = b$ e tomemos $x = x^0$. Então:

$$Ax^0 + (b - Ax^0)x_{n+1}^0 = b, \quad (2.22 \text{ a})$$

$$Ax^0(1 - x_{n+1}^0) = b(1 - x_{n+1}^0). \quad (2.22 \text{ b})$$

Mas, (2.22 b) só será verdadeira quando:

$$1 - x_{n+1}^0 = 0,$$

$$x_{n+1}^0 = 1. \quad (2.22 \text{ c})$$

Procedendo analogamente para $A^T y + (A^T y^0 + z^0 - c)y_{m+1}^0 + z = c$, obtemos:

$$y_{m+1}^0 = -1. \quad (2.23)$$

Das demais restrições obtemos:

$$x_{n+2}^0 = M_D - (A^T y^0 + z^0 - c)^T x^0, \quad (2.24)$$

$$z_{n+1}^0 = M_P - (b - Ax^0)^T y^0, \quad (2.25)$$

$$z_{n+2}^0 = 1. \quad (2.26)$$

A partir daqui, buscamos as soluções de (PLA) e (PDA) de maneira que se relacionam com os problemas originais (PL) e (PD), respectivamente, de acordo com o Teorema 3 abaixo.

Teorema 3 [FP – 1993]: Sejam x^* e (y^*, z^*) soluções ótimas dos problemas originais (PL) e (PD), respectivamente. E além disso em (2.21), suponha que:

$$M_D > (A^T y + z^0 - c)^T x^*,$$

e

$$M_P > (b - Ax^0)^T y^*.$$

Então as duas afirmações abaixo são verdadeiras:

- i. Uma solução viável $(\bar{x}, \bar{x}_{n+1}, \bar{x}_{n+2})$ de (PLA) é um ponto de mínimo se, e somente se $\bar{x}$ resolve (PL) e $\bar{x}_{n+1} = 0$.
- ii. Uma solução viável $(\bar{y}, \bar{y}_{m+1}, \bar{z}, \bar{z}_{n+1}, \bar{z}_{n+2})$ de (PDA) é um ponto de máximo se, e somente se $(\bar{y}, \bar{z})$ resolve (PD) e $\bar{y}_{m+1} = 0$.

A prova deste teorema encontra-se na referência citada.

Devemos observar que este é exatamente o *método do M-Grande*, tradicionalmente usado em associação com o Simplex e aqui adaptado para os métodos primais-duais. Outros valores para M_P e M_D foram sugeridos por pesquisadores como M. Kojima e Kevin A. McShane [Levkovitz – 1992].



2.5.2 Critério de Parada

Ao contrário do método Simplex, o algoritmo de Ponto Interior Primal-Dual nunca encontra uma solução exata para o problema de PL. Por isso, o algoritmo precisa usar um *critério de parada* para decidir quando a iteração corrente está próxima o suficiente da solução ótima [Wright - 1997].

Muitos algoritmos consideram uma boa solução aproximada a solução que possui os resíduos primal $r_b = Ax - b$, dual $r_c = A^T y + z - c$ e a métrica dual μ suficientemente pequenos. Para esse fim pode-se usar medidas relativas diminuindo os problemas de escala dos dados [Wright - 1997]. Testes típicos para uma solução (x, y, z) são:

$$\frac{\|r_b\|}{1 + \|b\|} = \frac{\|Ax - b\|}{1 + \|b\|} \leq \tau_1, \quad (2.27 \text{ a})$$

$$\frac{\|r_c\|}{1 + \|c\|} = \frac{\|A^T y + z - c\|}{1 + \|c\|} \leq \tau_1, \quad (2.27 \text{ b})$$

$$\frac{|c^T x - b^T y|}{1 + |c^T x|} \leq \tau_1, \quad (2.27 \text{ c})$$

onde τ_1 é a tolerância. Naturalmente outros critérios podem ser adotados em concordância com a aplicação a um problema específico. Para outros tipos de critério de parada, veja [FP - 1993] e [BM - 1992].

2.5.3 Algoritmo Primal-Dual

Como último tópico deste capítulo, daremos uma estrutura mais “completa” do algoritmo primal-dual para uso no estudo dos próximos capítulos.



Fase I

Passo 1: (*Inicialização*). Faça $k = 0$, encontre $(x^0, y^0, z^0) \in \Omega^0$ uma solução inicial e defina $\tau_1 > 0$ uma tolerância suficientemente pequena.

Fase II

Passo 2: (*Verificação da otimalidade*). Se

$$\frac{\|r_b\|}{1 + \|b\|} = \frac{\|Ax - b\|}{1 + \|b\|} \leq \tau_1,$$

$$\frac{\|r_c\|}{1 + \|c\|} = \frac{\|A^T y + z - c\|}{1 + \|c\|} \leq \tau_1,$$

$$\frac{|c^T x - b^T y|}{1 + |c^T x|} \leq \tau_1,$$

então **pare**. A solução encontrada é ótima.

Passo 3: (*Cálculo da direção do passo*). Seja $\sigma_k \in [0, 1]$ e $\mu_k = \frac{(x^k)^T z^k}{n}$. Calcule:

$$AZ_k^{-1} X_k A^T \Delta y^k = A(X_k e - \sigma_k \mu_k Z_k^{-1} e)$$

$$\Delta z^k = -A^T \Delta y^k \quad (2.10)$$

$$\Delta x^k = -X_k e + \sigma_k \mu_k Z_k^{-1} e - Z_k^{-1} X_k \Delta z^k$$

Passo 4: (*Atualização*). Faça

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta z^k).$$

Faça $k = k + 1$ e volte ao passo 2.

No próximo capítulo, discutiremos como obter um plano de corte a partir de uma solução do método de Pontos Interiores e os ajustes necessários no método de Pontos Interiores para a sua utilização no método de Planos de Cortes.

Capítulo

3

Planos de Corte e Pontos Interiores

O *Método de Planos de Corte* foi a primeira técnica sistemática desenvolvida para resolver problemas de Programação Inteira. A técnica é simples: é construída uma sequência de problemas lineares pela adição sucessiva de inequações lineares que eliminem a solução ótima do problema linear anterior mas que não eliminem nenhum ponto viável inteiro. Este método foi primeiro proposto por Dantzig *et alii* (1954) para resolução do problema do caixeiro viajante [Rangel – 1995], mas ganhou notoriedade através do trabalho de Gomory [Gomory - 1958].

A idéia de Gomory entusiasmou muito a comunidade científica em princípio, mas acabou sendo considerada de pouca utilidade prática. Porém, mais recentemente outras maneiras de gerar os cortes, em conjunto com desenvolvimentos mais recentes em Programação Linear e Inteira reacenderam o interesse nos planos de cortes.

Em geral os problemas lineares envolvidos no método de planos de cortes são resolvidos usando o método Simplex. O propósito deste capítulo é o de mostrar como resolver estes problemas lineares usando o método de Pontos Interiores, como gerar planos de cortes (de Gomory), reotimizar e recuperar a viabilidade do problema após a adição de planos de cortes .

3.1 Passos Iniciais

Antes de prosseguirmos, vejamos algumas definições úteis.

Considere o problema de Programação Linear Inteira:

$$\begin{array}{ll} \text{Min.} & c^T x \\ \text{S.a} & Ax = b \\ (\text{PI}) & x \geq 0 \\ & x \text{ inteiro,} \end{array}$$

com $A \in \mathbf{Z}^{m \times n}$, $b \in \mathbf{Z}^m$ e $x, c \in \mathbf{Z}^n$, onde $\mathbf{Z}$ é o conjunto dos números inteiros e $P^I = \{x \in \mathbf{Z}^n / Ax = b, x \geq 0\}$, o conjunto das soluções viáveis de (PI).



O problema relaxado de (PI) (chamado de *relaxação linear*) é o problema de Programação Linear obtido de (PI) eliminando a restrição de que a solução seja inteira, isto é:

$$\begin{array}{ll} \text{Min. } c^T x \\ \text{(PL)} \quad \text{S.a } Ax = b \\ \quad \quad x \geq 0. \end{array}$$

O problema *dual* da relaxação de (PL) é:

$$\begin{array}{ll} \text{Max. } b^T y \\ \text{(PD)} \quad \text{S.a } A^T y + z = c \\ \quad \quad z \geq 0, \quad y \text{ irrestrito.} \end{array}$$

Denominamos *inequação válida* para (PI) a uma inequação linear que é satisfeita por todos os pontos viáveis do (PI), isto é:

$$a_0^T x \leq b_0, \quad \forall x \in P^I,$$

com $a_0 \in \mathbb{R}^n$ e $b_0 \in \mathbb{R}$.

Quando uma inequação válida é adicionada a (PL) e ela elimina um ponto fracionário x^* de (PL), ou seja,

$$a_0^T x^* > b_0,$$

a chamamos de *plano de corte*.

Naturalmente a inclusão de uma inequação a um (PL) tem uma série de implicações a serem esclarecidas, por exemplo: como gerar inequações válidas, como reotimizar o problema após a adição e/ou eliminação de inequações, quando devemos gerar novos cortes e quando podemos eliminar cortes que tornam-se desnecessários para o problema. Mas antes é instrutivo termos uma primeira visão de um algoritmo de Planos de Cortes.

Algoritmo de Planos de Corte [FW – 1996]

Passo 1: (*Resolvendo a relaxação*). Resolva a relaxação (PL) correspondente ao problema (PI). Se o problema (PL) for inviável ou ilimitado, então **pare**, (PI) é inviável ou ilimitado. Se x^* for inteiro, então **pare**, resolvemos o (PI). Caso contrário, tome x^* uma solução ótima de (PL).

Passo 2: (*Problema de Separação*). Encontre uma inequação válida com as seguintes propriedades:

$$a_0^T x^* > b_0 \text{ e } a_0^T x \leq b_0, \quad \forall x \in P^I.$$



Passo 3: (*Adicionando cortes*). Acrescente esta nova inequação ao conjunto de restrições e volte ao passo 1.

3.2 Gerando Planos de Cortes de Gomory

Podemos gerar diferentes inequações válidas (planos de cortes) satisfazendo as condições do passo 2 do algoritmo de planos de cortes (seção 3.1). Nesta seção abordaremos o *Corte de Gomory*, tendo por base o trabalho de Gomory [Gomory - 1958] e Ferreira e Wakabayashi [FW - 1996].

O nome *plano de corte* vem do fato de que a cada iteração a solução ótima de (PL) corrente vai sendo *cortada*. Quando resolvemos a relaxação de (PI), ou seja, resolvemos (PL) usando o algoritmo Simplex, podemos gerar cortes a partir de qualquer restrição associada à variável básica que tenha valor fracionário; e/ou da função objetivo, usando os custos reduzidos ótimos, quando estes têm valor fracionário.

Seja uma partição da matriz $A = [B | N]$, tal que a submatriz B é uma matriz não-singular correspondente às variáveis básicas x_B e N a submatriz correspondente às variáveis não-básicas x_N . O valor da solução ótima do (PL) será denotada por c^* , ou seja, $c^* = c^T x^*$, $\bar{A} = B^{-1}N$, $\bar{b} = B^{-1}b$ e $\bar{c}^T = c_N^T - c_B^T \bar{A}$.

Uma solução básica, relativa à base B , pode ser escrita na forma:

$$x_B = B^{-1}b - B^{-1}Nx_N = \bar{b} - \bar{A}x_N.$$

Ou seja, considerando a i -ésima variável básica com valor fracionário temos:

$$\bar{b}_i - \sum_{j \in N} \bar{a}_{ij} x_j = x_{B_i}, \quad \forall i \in B. \quad (3.1)$$

Considere a função $f_a = a - [a]$, onde $[a]$ (função piso) representa o maior inteiro menor ou igual do que a , e a função f_a representa a parte fracionária de a .

Uma inequação válida para (PI) pode ser obtida considerando que como $x \geq 0$, podemos tomar a parte inteira do lado esquerdo da equação (3.1) e obter:

$$[\bar{b}_i] - \sum_{j \in N} [\bar{a}_{ij}] x_j \leq x_{B_i}, \quad \forall i \in B. \quad (3.2)$$

Subtraindo (3.2) de (3.1), obtemos:

$$f_{\bar{b}_i} - \sum_{j \in N} f_{\bar{a}_{ij}} x_j \leq 0, \quad \forall i \in B. \quad (3.3)$$

Rescrevendo (3.3), obtemos:

$$\sum_{j \in N} f_{\bar{a}_i} x_j \geq f_{\bar{b}_i}, \quad \forall i \in B. \quad (3.4)$$

A inequação (3.4) é um plano de corte, mais precisamente, um plano de corte de Gomory gerado de variáveis básicas com valores fracionários.

Uma outra maneira de se gerar cortes é usando a função objetivo.

Rescrevendo a função objetivo temos:

$$\begin{aligned} c_B^T \bar{b} + \bar{c}_N^T x_N &= c^T x, \\ c_B^T \bar{b} + \sum_{j \in N} \bar{c}_j x_j &= c^T x. \end{aligned} \quad (3.5)$$

Analogamente aos passos (3.1) – (3.4) acima, podemos obter um corte através da função objetivo de (PL).

Sendo $x \geq 0$, tomemos a parte inteira do lado esquerdo da equação (3.5) obtendo a seguinte inequação:

$$\lfloor c_B^T \bar{b} \rfloor + \sum_{j \in N} \lfloor \bar{c}_j \rfloor x_j \leq c^T x. \quad (3.6)$$

Substituindo (3.5) em (3.6), temos:

$$\begin{aligned} \lfloor c_B^T \bar{b} \rfloor + \sum_{j \in N} \lfloor \bar{c}_j \rfloor x_j &\leq c_B^T \bar{b} + \sum_{j \in N} \bar{c}_j x_j, \\ (\lfloor c_B^T \bar{b} \rfloor - c_B^T \bar{b}) + \sum_{j \in N} (\lfloor \bar{c}_j \rfloor - \bar{c}_j) x_j &\leq 0, \\ -f_{c_B^T \bar{b}} - \sum_{j \in N} f_{\bar{c}_j} x_j &\leq 0. \end{aligned} \quad (3.7)$$

Multiplicando (3.7) por (-1) , obtemos:

$$f_{c_B^T \bar{b}} + \sum_{j \in N} f_{\bar{c}_j} x_j \geq 0. \quad (3.7 \text{ a})$$

Observe que a inequação acima (3.7 a) tem que ser estritamente maior que zero, pois só geramos planos de cortes a partir dos custos relativos fracionários e a soma das partes fracionárias nunca será igual a zero, portanto rescrevemos tal inequação como segue:

$$f_{c_B^T \bar{b}} + \sum_{j \in N} f_{\bar{c}_j} x_j > 0 \quad (3.8)$$

A validade das inequações (3.4) e (3.8) para (PI) pode ser resumida no Teorema 4, abaixo, adaptado de [FW - 1996].



Teorema 4: Considere o problema (PI), onde A e b são inteiros. Seja x^* uma solução ótima de (PL) e B a base associada à solução x^* . Então as inequações:

$$\sum_{j \in N} f_{\bar{a}_{ij}} x_j \geq f_{\bar{b}_i}, \quad \forall i \in B,$$

$$f_{c_B^T \bar{b}} + \sum_{j \in N} f_{\bar{c}_{Nj}} x_j > 0,$$

são planos de cortes para a solução x^* , caso $\bar{b}_i$ ou $\bar{c}^T$ sejam fracionários, respectivamente.

Os planos de cortes do teorema 4 são conhecidos como *cortes de Gomory*. Em geral, estes cortes são gerados a partir do método Simplex, porém Mitchell [Mitchell - 1997] mostrou que é possível gerar cortes de Gomory usando o método de Pontos Interiores, o que será motivo de estudo na próxima subseção.

3.3 Planos de Cortes Gerados a partir de Pontos Interiores

Até aqui vimos como gerar cortes a partir das restrições e/ou da função objetivo do problema quando estes têm valores fracionários e são obtidos através do método Simplex. Apresentaremos aqui como gerar estes cortes quando trabalhando com Método de Pontos Interiores.

Quando usamos um método de Pontos Interiores para resolver a relaxação (PL) é necessário calcular, em cada iteração, a seguinte matriz definida positiva:

$$(AD^2A^T),$$

onde $D = X^{1/2}Z^{-1/2}$, é uma matriz diagonal $n \times n$, com $X_{ii} = x_i$ e $Z_{ii} = z_i$ matrizes diagonais $n \times n$ (veja seção 2.2). Outros valores podem ser atribuídos a D , tais como X ou Z^{-1} , dependendo do método de Ponto Interior utilizado; aqui estamos supondo o uso do método de Ponto Interior Primal-Dual discutido no capítulo 2.

Esta matriz (AD^2A^T) pode ser usada para gerar um plano de corte. Este corte é equivalente ao plano de corte de Gomory se a relaxação corrente for resolvida até a otimalidade e for não-degenerada.

Mitchell [Mitchell - 1997] propõe o uso da matriz M^{cut} para gerar os planos de cortes:

$$M^{cut} := D^2A^T(AD^2A^T)^{-1}A.$$



Esta matriz é facilmente construída, pois a cada iteração do método de Ponto Interior temos a decomposição de Cholesky de AD^2A^T . As linhas da matriz M^{cut} podem ser geradas uma a uma e servirão para a criação dos cortes. A i -ésima linha de M^{cut} , d_i , pode ser obtida resolvendo-se o sistema:

$$(AD^2A^T)v_i = D_{ii}^2a_i, \quad (3.9)$$

onde:

a_i – é a i -ésima coluna de A ;

$D_{ii}^2a_i$ – é a i -ésima linha de D^2A^T .

Portanto:

$$d_i^T = v_i^T A.$$

O Teorema 5 a seguir [Mitchell - 1997] relaciona a otimalidade de (PL) com a solução do método Simplex e com a matriz M^{cut} .

Teorema 5: Suponha que (PL) seja não degenerado e que x_i^* é uma variável básica em sua solução ótima. Conforme avançamos nas iterações de Pontos Interiores, a i -ésima linha de M^{cut} tende à linha correspondente à variável básica x_i^* da solução ótima obtida pelo método Simplex.

Prova: Particionaremos a matriz A em colunas básicas B , e não básicas N , e assumiremos, sem perda de generalidade, que as primeiras m colunas de A serão as colunas das variáveis básicas, tal que $A = [B \mid N]$. Definiremos também como D_B e D_N as matrizes relativas à partição da matriz D correspondente a B e N , respectivamente. Então, na solução ótima temos que $D_N = 0$, pois $D = X^{1/2}Z^{-1/2}$, onde $X_{ii} = x_i$ e $Z_{ii} = z_i$ são matrizes diagonais $n \times n$.

Rescrevendo a matriz M^{cut} , de acordo com as partições acima, teremos:

$$M^{cut} = \begin{bmatrix} D_B^2 & 0 \\ 0 & D_N^2 \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} \left(\begin{bmatrix} B & N \end{bmatrix} \begin{bmatrix} D_B^2 & 0 \\ 0 & D_N^2 \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} \right)^{-1} \begin{bmatrix} B & N \end{bmatrix}.$$

Como $D_N = 0$, temos:

$$M^{cut} = \begin{bmatrix} D_B^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} \left(\begin{bmatrix} B & N \end{bmatrix} \begin{bmatrix} D_B^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} \right)^{-1} \begin{bmatrix} B & N \end{bmatrix}$$

$$M^{cut} = \begin{bmatrix} D_B^2 B^T \\ 0 \end{bmatrix} \left(\begin{bmatrix} B D_B^2 & 0 \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} \right)^{-1} \begin{bmatrix} B & N \end{bmatrix}$$

$$M^{cut} = \begin{bmatrix} D_B^2 B^T \\ 0 \end{bmatrix} (B D_B^2 B^T)^{-1} \begin{bmatrix} B & N \end{bmatrix} = \begin{bmatrix} B^{-1} \\ 0 \end{bmatrix} \begin{bmatrix} B & N \end{bmatrix},$$

e finalmente:

$$M^{cut} = \begin{bmatrix} I & B^{-1}N \\ 0 & 0 \end{bmatrix}.$$

Isto é, as m primeiras linhas de M^{cut} nos dão o quadro Simplex ótimo do (PL). ■

Para gerarmos o corte, partimos da relação $d_i^T = v_i^T A$, onde v_i é dado por (3.9).

Multiplicando-se por x em ambos os lados temos

$$d_i^T x = v_i^T Ax,$$

lembrando que $Ax = b$, então a equação acima fica:

$$d_i^T x = v_i^T b. \quad (3.10)$$

Podemos usar esta equação para gerar inequações válidas para o (PI) em qualquer iteração. No limite da otimalidade esta inequação será o corte de Gomory associado à variável básica x_i . Este resultado está enunciado no Teorema 6 [Mitchell - 1997] abaixo.

Teorema 6: Seja d_i a i -ésima linha da matriz M^{cut} e seja $v_i = D_{ii}^2 (AD^2 A^T)^{-1} a_i$. Então a inequação

$$f_{d_i^T} x \geq f_{v_i^T b} \quad (3.11)$$

é satisfeita para toda solução viável de (PI). Se (PL) é não degenerado então esta restrição dá o corte de Gomory para a variável x_i quando $(c^T \bar{x} - b^T \bar{y}) \rightarrow 0$.

Prova: Temos que:

$$d_i^T x = v_i^T b, \quad (3.12)$$

para cada x satisfazendo $Ax = b$. Então podemos tomar a parte inteira de (3.12), obtendo a inequação:

$$\lfloor d_i^T \rfloor x \leq v_i^T b. \quad (3.13)$$

Considerando que, para pontos inteiros viáveis, o valor do lado esquerdo da inequação (3.13) deve ser um inteiro, podemos também tomar a parte inteira do lado direito, obtendo:



$$\lfloor d_i^T \rfloor x \leq \lfloor v_i^T \rfloor b. \quad (3.14)$$

Subtraindo (3.14) de (3.12), obtemos:

$$f_{d_i^T} x \geq f_{v_i^T} b \quad (3.15)$$

Na otimalidade a i -ésima componente de d_i tende a 1, e se o problema é não degenerado, então d_i nos dá exatamente a linha correspondente a variável básica x_i do método Simplex. E ainda mais, o valor de x_i é $v_i^T b$. ■

Para gerar cortes de Gomory a partir da função objetivo partimos da propriedade de que quando usamos um método de Pontos Interiores, as folgas duais estão disponíveis e são iguais aos custos reduzidos das variáveis primais na otimalidade.

Como vimos na seção 3.2, se o valor ótimo da função objetivo, $c^* = c^T x^*$, é fracionário, então a inequação:

$$f_{c^T \bar{b}} + \sum_{j \in N} f_{\bar{c}_j} x_j > 0,$$

é válida para o (PI) mas é violada pela solução ótima de (PL), servindo portanto como um plano de corte. É possível gerar esta inequação através de qualquer par de soluções primal-dual viáveis. O Teorema 7 [Mitchell - 1997] abaixo nos mostra como obtê-la.

Teorema 7: Sejam $\bar{x}$ e $\bar{y}$ soluções primal e dual viáveis respectivamente e $\bar{z}$ a folga dual correspondente. Suponha que existe um inteiro p tal que $p < b^T \bar{y} \leq c^T \bar{x} < p+1$. Então a restrição:

$$f_{b^T \bar{y}} + \sum_{i=1}^n f_{\bar{z}_i} x_i \geq 1 \quad (3.16)$$

é válida para (PI) mas violada por $\bar{x}$. Mais ainda: se o (PL) for não-degenerado, então esta restrição é equivalente ao corte de Gomory (3.8) no limite $(c^T \bar{x} - b^T \bar{y}) \rightarrow 0$.

Prova: Vamos mostrar inicialmente que a inequação (3.16) é válida para toda solução de (PI). De (PD) segue:

$$c^T \bar{x} = b^T \bar{y} + \bar{z}^T \bar{x} \quad (3.17 a)$$

$$c^T \bar{x} = \lfloor b^T \bar{y} \rfloor + f_{b^T \bar{y}} + \lfloor \bar{z}^T \bar{x} \rfloor + f_{\bar{z}^T \bar{x}} \quad (3.17 b)$$



$$c^T \bar{x} = \lfloor b^T \bar{y} \rfloor + \lfloor \bar{z}^T \bar{x} \rfloor + f_{b^T \bar{y}} + f_{\bar{z}^T \bar{x}} \quad (3.17 \text{ c})$$

Seja x^* a solução ótima do (PI). Como $c^T x^*$ é inteiro, $f_{b^T \bar{y}} + f_{\bar{z}^T \bar{x}}$ tem de ser inteiro e seu valor tem de ser no mínimo um, pois $f_{b^T \bar{y}} > 0$. Logo:

$$f_{b^T \bar{y}} + \sum_{i=1}^n f_{\bar{z}_i} x_i \geq 1.$$

Agora vamos mostrar que o ponto $\bar{x}$ viola a inequação (3.16).

Seja:

$$b^T \bar{y} > p, \quad (3.18 \text{ a})$$

$$b^T \bar{y} < p+1. \quad (3.18 \text{ b})$$

Se tomarmos o piso de $b^T \bar{y}$ as inequações de (3.18) serão:

$$\lfloor b^T \bar{y} \rfloor = p, \quad (3.19 \text{ a})$$

$$\lfloor b^T \bar{y} \rfloor < p+1. \quad (3.19 \text{ b})$$

Logo:

$$c^T x < p+1 \Rightarrow c^T x - p < 1 \quad (3.20)$$

Substituindo (3.19 a) em (3.20), temos:

$$c^T \bar{x} - \lfloor b^T \bar{y} \rfloor < 1. \quad (3.21)$$

De (3.17 b) temos que:

$$c^T \bar{x} - \lfloor b^T \bar{y} \rfloor = f_{b^T \bar{y}} + \bar{z}^T \bar{x}. \quad (3.22)$$

Substituindo (3.22) em (3.21) obtemos:

$$f_{b^T \bar{y}} + \bar{z}^T \bar{x} < 1. \quad (3.23)$$

Considerando apenas a parte fracionária de $\bar{z}^T \bar{x}$, temos:

$$f_{b^T \bar{y}} + f_{\bar{z}^T \bar{x}} \leq f_{b^T \bar{y}} + \bar{z}^T \bar{x} < 1.$$

Portanto, o ponto $\bar{x}$ viola a inequação (3.16).

Finalmente, se o (PL) é não degenerado e se $c^T \bar{x} = b^T \bar{y}$, então $\bar{z}_i$ é igual aos custos reduzidos da variável $\bar{x}_i$, isto é $\bar{z}_i = \bar{c}_i$. Segue que o corte definido em (3.16) é o corte de Gomory. ■

3.4 Reotimização

Discutiremos agora algumas técnicas importantes do uso do método de Pontos Interiores com Planos de Cortes, tais como: a reiniciação do algoritmo e recuperação da viabilidade do problema após a adição de planos de cortes – *reiniciando e recuperando a viabilidade primal*, a importância da técnica de *terminação antecipada – resolvendo a relaxação aproximadamente*, e a eliminação de inequações desnecessárias do conjunto de restrições – *gerando e eliminando restrições*.

Reiniciando o Algoritmo e Recuperando a Viabilidade Primal

Um ponto importante no uso de Planos de Cortes com método de Pontos Interiores é como recuperar a viabilidade do problema após a adição de novos cortes e reotimizá-lo.

Suponha que o plano de corte $a_0^T \tilde{x} \leq b_0$ tenha sido adicionado ao problema, já com as variáveis de folga ($a_0^T x = b_0$). Então o problema tem agora a forma:

$$\begin{array}{ll}
 \text{Min. } c^T x & \\
 \text{S.a } Ax = b & \\
 a_0^T x = b_0 & \\
 x \geq 0 &
 \end{array}
 \quad (\text{PL2})$$

com dual correspondente:

$$\begin{array}{ll}
 \text{Max. } b^T y + b_0 y_0 & \\
 \text{S.a } A^T y + a_0 y_0 + z = c & \\
 z \geq 0, \quad y, y_0 \text{ irrestrito.} &
 \end{array}
 \quad (\text{PD2})$$

Sabemos que a solução corrente $\hat{x}$ satisfaz $Ax = b$ mas não satisfaz $a_0^T x = b_0$. O ponto dual corrente $\hat{y}, \hat{z}$ é viável para (PD2) se tomarmos $y_0 = 0$. Temos também que $\hat{x} > 0, \hat{y} > 0$ e as folgas primal e dual são estritamente positivas, com exceção da folga associada à nova restrição. Reiniciar o problema deste ponto pode se tornar caro, pois como $y_0 = 0$ não é um ponto interior, o método de Pontos Interiores tentará primeiro gerar direções para o centro do politopo e depois direções que levem à solução. Uma maneira de se contornar este problema é tomando $y_0 = \varepsilon$ (tornando o dual inviável) e resolver o problema (PL2) usando um método de Ponto Interior Inviável. Também é benéfico levarmos a ε qualquer variável com valor menor do que ε .

Um certo cuidado pode ser tomado ao adotar este procedimento, pois ao tomando $y_0 = \varepsilon$ tornamos também o dual inviável. Os parâmetros de controle do método de Pontos Interiores são bem calculados para que haja um balanço entre a busca pela otimalidade e pela viabilidade. Sem este cuidado, o algoritmo pode acabar longe da otimalidade enquanto busca pela viabilidade.

Se a solução ótima $\hat{x}$ de (PL2) satisfaz a restrição adicionada na igualdade, então $\bar{y}_0$ será estritamente positivo (devido ao fato de que a maioria dos métodos de Pontos Interiores satisfaz a complementaridade estrita – seção 2.2). Assim, o algoritmo aumenta y_0 de ε para um número positivo, distanciando-se da face $y_0 = 0$.

Mais recentemente [Gondzio - 1998], foram desenvolvidas novas estratégias baseadas em *Centros Analíticos de Polítopos* onde a viabilidade – primal e dual – é recuperada mais rapidamente, acelerando de modo geral o cálculo da solução do problema, mas estas novas técnicas fogem do escopo deste trabalho.

Resolvendo a Relaxação Aproximadamente

Uma técnica que torna prático um algoritmo de Planos de Cortes e Pontos Interiores é a *terminação antecipada*: a relaxação corrente é resolvida aproximadamente. Ou seja, a solução da relaxação corrente linear é interrompida antes que a solução ótima seja obtida e planos de cortes são introduzidos para eliminar a solução aproximada.

Mitchell [Mitchell - 1995] faz algumas considerações a respeito das vantagens e desvantagens deste procedimento. As principais vantagens são:

- Na terminação antecipada, as soluções iniciais para o próximo subproblema são mais centradas; assim, poucas iterações são requeridas para resolver o próximo subproblema.
- Poucas iterações são requeridas no problema corrente.
- A solução antecipada está mais próxima do centro da região viável de (PL) do que da solução ótima, assim o algoritmo é capaz de encontrar cortes mais *profundos*, isto é, cortes que eliminam mais a região viável de (PL) e aproximam-se mais rapidamente do envoltório convexo de (PI).
- Pode ocorrer que alguns cortes encontrados pela busca antecipada não sejam encontrados até a próxima relaxação, se a relaxação tiver sido resolvida até a otimalidade.

E as duas principais desvantagens são:



- Podemos não encontrar planos de corte: só há a garantia de encontrá-los se resolvêssemos até a otimalidade. Assim, gastamos tempo procurando antecipadamente.
- Inversamente, pode ser que se resolvêssemos a relaxação corrente até a otimalidade, a solução ótima de (PL) seria também solução do (PI); no entanto, usando a terminação antecipada, a solução na iteração corrente pode estar fora do envoltório convexo da região viável, e assim a rotina de separação retorna cortes desnecessários.

Para balancear as vantagens e desvantagens desta implementação ele usa uma tolerância que se altera dinamicamente sobre o gap dual $(c^T x - b^T y)$: somente procuramos por planos de cortes se a tolerância é menor que o gap dual. Assim, aumentamos o valor da tolerância quando encontramos muitos planos de cortes e, diminuimos quando poucos (ou nenhum) planos de cortes forem encontrados.

A terminação antecipada, quando bem dimensionada, serve para tornar o algoritmo competitivo em relação ao Simplex com planos de corte. Este fato nos leva a perguntar se a mesma idéia de terminar antecipadamente não poderia ser utilizada no próprio Simplex. Mitchell [Mitchell - 1995] apresenta algumas das razões pelas quais isto não é apropriado. Em primeiro lugar, o Simplex anda por pontos extremos, e por isso é provável que muitas variáveis sejam não fracionárias (na verdade, nulas), e elementos vitais podem ser pivotados na base exatamente nas últimas iterações. Em segundo lugar: pode acontecer de a solução da relaxação corrente resolver o (PI), mas a terminação antecipada pode violar restrições mesmo que a solução ótima não o faça; isso levaria a uma adição de planos de corte desnecessários. E em terceiro lugar: temos a comparação de tempo entre a procura por planos de corte violados pelo método de Pontos Interiores e pelo método Simplex. Como as rotinas de separação são mais “caras” que as iterações do Simplex, torna-se impraticável fazer a busca por planos de corte em cada iteração do Simplex.

Gerando e Eliminando Restrições

Na seção 3.2 estivemos preocupados em *como* gerar os planos de cortes, porém quando executamos a rotina para gerar tais cortes podemos encontrar diversas inequações violadas. A adição destas restrições muda o problema para



$$\begin{array}{ll}
 & \text{Min. } c^T x \\
 & \text{S.a } Ax = b \\
 (\text{PL1}) & A_0^T x = b_0 \\
 & x \geq 0
 \end{array}$$

com dual

$$\begin{array}{ll}
 & \text{Max. } b^T y + b_0 y_0 \\
 (\text{PD1}) & \text{S.a } A^T y + A_0 y_0 + z = c \\
 & z \geq 0, \quad y, y_0 \text{ irrestrito}
 \end{array}$$

com A_0 a matriz de restrições adicionadas.

Admitindo uma “pequena” inviabilidade dual se para cada componente de y_0 atribuímos uma pequena quantia positiva, ε , reotimizamos o problema usando, por exemplo, o método de Pontos Interiores Inviável. O fato é que a quantidade de restrições adicionadas pode ser muito grande, o que traz algumas implicações, pois o grande número de restrições adicionadas pode mudar tanto o problema relaxado (PL) e (PD), quanto a resolução do problema assim modificado, tornando caro o processo de solução, tanto do ponto de vista do número de iterações do método de Pontos Interiores, quanto do de algoritmos numéricos (decomposição de Cholesky, resolução de sistemas lineares, etc). Desse modo é aconselhável adicionar apenas um subconjunto dos cortes encontrados, ou eliminar inequações inseridas anteriormente que tornam-se desnecessárias para o problema.

Um critério sugerido por Padberg e Rinaldi [PR - 1991], tomado inicialmente, é verificar se o corte recém gerado já foi gerado em outras iterações, evitando inequações duplicadas. Outro critério não tão imediato, é calcular o tamanho absoluto da violação na solução primal corrente $\bar{x}$, ou seja, $|a_0^T \bar{x} - b_0|$. Quanto maior este número, mais *profundo* o corte [PR - 1991]. Os cortes são assim classificados e do conjunto resultante de cortes, são usados apenas os p primeiros, onde p é um parâmetro pré-estabelecido.

3.5 Planos de Corte com Pontos Interiores

Embora já tenhamos abordado uma estrutura para o algoritmo de planos de cortes na seção 3.1, vamos aqui elaborar um algoritmo com base na discussão feita até aqui. Porém, antes, lembremos que $\Omega^0 = \{(x, y, z) / Ax = b, A^T y + z = c, (x, z) > 0\}$ e $gap = c^T x - b^T y$.

Algoritmo de Planos de Cortes com Pontos Interiores

Passo 1: (*Inicialização*). Encontre $x, y, z \in \Omega^0$, defina as tolerâncias τ_1 (otimalidade), $\tau_2 > \tau_1$ (geração de cortes) e $\varepsilon > 0$.

Passo 2: (*Resolvendo aproximadamente a relaxação*). Execute uma iteração do método de Pontos Interiores escolhido.

(a) Se $gap < \tau_1$, tome x^* a sua solução. Vá ao passo 7.

(b) Se $gap < \tau_2$, vá ao passo 3.

(c) Caso contrário, repita este passo.

Passo 3: (*Procurando planos de corte*). Tente gerar planos de cortes. Se nenhum for gerado, reduza τ_2 e volte ao Passo 2; caso contrário, vá ao passo 4.

Passo 4: (*Adicionando planos de corte*). Do conjunto de cortes encontrados, selecione um subconjunto para adição ao problema. Se este subconjunto é muito grande aumente τ_2 . Se for muito pequeno, diminua τ_2 .

Passo 5: (*Eliminando planos de corte*). Elimine restrições desnecessárias.

Passo 6: (*Modificando a iteração corrente*). Faça as componentes x , y e z da solução menores do que ε iguais a ε . Volte ao passo 2.

Passo 7: (*Finalize*). Se x^* for inteiro, o problema (PI) foi resolvido. Caso contrário, (PL) não tem solução inteira.

Em [Mitchell - 1997b], o autor relata algumas experiências computacionais do uso do método de Pontos Interiores em método de Planos de Cortes, para a resolução de problemas clássicos da literatura, entre eles o Problema de Ordenação Linear que discutiremos com maiores detalhes no capítulo 5.

Na seção seguinte, falaremos sobre o uso do método de Pontos Interiores na resolução dos subproblemas gerados pelos métodos Branch and Bound e Branch and Cut.



Capítulo

4

Branch and Cut e Método de Pontos Interiores

Um outro método clássico na resolução de problemas de Programação Inteira (PI) é o *Método Branch and Bound* (BB).

Considere um problema de Programação Inteira:

$$\begin{array}{ll}
 \text{Min. } & c^T x \\
 \text{(PI)} \quad \text{S.a } & Ax = b \\
 & x \geq 0 \\
 & x \text{ inteiro,}
 \end{array}$$

com $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ e $x, c \in \mathbb{Z}^n$. O problema relaxado de (PI) – *relaxação linear* – é o problema de Programação Linear obtido de (PI) retirando-se a restrição de que as variáveis devem ser inteiras, isto é:

$$\begin{array}{ll}
 \text{Min. } & c^T x \\
 \text{(PL)} \quad \text{S.a } & Ax = b \\
 & x \geq 0.
 \end{array}$$

O método BB é conceitualmente simples: primeiro resolvemos a relaxação (PL) correspondente ao (PI). Se a sua solução for inviável ou ilimitada, o (PI) também o é. Se a solução for inteira, resolvemos o (PI). Caso contrário, o problema é particionado em subproblemas disjuntos. Em seguida, selecionamos um dos subproblemas e repetimos o processo. A solução da relaxação linear é usada para calcular limites para o valor da função



objetivo. Com estes limites podemos descartar ou criar novos subproblemas num processo finito que converge à solução do (PI), se houver.

Este método pode ser esquematizado por uma árvore de busca cuja raiz representa o problema original e os demais nós os subproblemas criados a partir das partições do problema relaxado.

O primeiro algoritmo BB foi proposto por Land e Doig em 1960 [LD - 1960]. Em 1963 o método tornou-se popular depois de sua aplicação, por Little *et alii* [LMSK - 1963], para resolver o problema do caixeiro viajante. O método BB passou então a ser utilizado para resolver problemas de (PI) e é usado como base em muitos softwares comerciais [Rangel - 1995].

Neste capítulo, além de descrevermos com maiores detalhes o método Branch and Bound, abordaremos também um método híbrido que combina o método de Planos de Cortes e o método Branch and Bound, chamado de *Método Branch and Cut* (BC). Em geral os subproblemas criados no método BB e BC são resolvidos através do método Simplex. Neste capítulo mostraremos como usar o método de Pontos Interiores para resolver estes subproblemas, como obter um ponto de partida para o próximo subproblema e o uso da técnica de terminação antecipada, onde se resolve a relaxação corrente *aproximadamente*.

4.1 O Método Branch and Bound

Como mencionamos no início deste capítulo, o método BB (também conhecido como enumeração implícita) é representado por uma árvore de busca. Cada nó da árvore representa um subproblema e a raiz está associada à relaxação (PL) de (PI).

Os conjuntos de soluções de (PI) e (PL) são

$$P' = \{x \in \mathbb{Z}^n / Ax = b, x \geq 0\} \text{ e}$$

$$P = \{x \in \mathbb{R}^n / Ax = b, x \geq 0\},$$



respectivamente. Dizemos que $P_j, j = 1, 2, \dots, k$ é uma divisão de P' se $\bigcup_{j=1}^k P_j = P'$. Uma divisão será chamada de *partição* quando $P_l \cap P_j = \emptyset \quad \forall l, j = 1, 2, \dots, k$ e $l \neq j$. O subproblema associado ao nó j da árvore de enumeração é da forma:

$$\begin{array}{ll} \text{Min.} & c^T x \\ \text{S. a} & x \in P_j \end{array}$$

Denotaremos por $\underline{c}_j = c^T \bar{x}^j$ o valor da função objetivo calculada na solução do subproblema j ($j = 1, 2, \dots, k$), este valor será usado como limite inferior para (PI) e por $\bar{c}$ um limite superior de (PI) obtido através de alguma solução inteira viável.

Divisão

Suponha que a divisão de (PI) é feita pela adição de restrições lineares. Uma maneira de fazer esta divisão é tomando-se $P' = P_1 \cup P_2$ com $P_1 = P' \cap \{x \in \mathcal{R}_+^n / d^T x \leq d_0\}$ e $P_2 = P' \cap \{x \in \mathcal{R}_+^n / d^T x \geq d_0 + 1\}$ sendo $d \in \mathbb{Z}^n$ e $d_0 \in \mathbb{Z}$ escolhidos de maneira adequada para que $\bar{x} \notin P_1 \cup P_2$ se $\bar{x}$ é a solução ótima de (PL), e mais, que a partição $P' = P_1 \cup P_2$ não elimine nenhum ponto de (PI). Com isso temos a possibilidade de que $\underline{c}_j > \underline{c}^0$ para $j = 1, 2$ e $\underline{c}^0$ é o valor da função objetivo da relaxação de (PI).

Na prática usamos apenas escolhas especiais para d e d_0 . Aqui estamos interessados na escolha de variáveis *dicotômicas*, onde $d = e_i$ para algum $i \in N$ e $d_0 = \lfloor \bar{x}_i \rfloor$. Então se $\bar{x}_i$ é fracionário, esta variável será inviável na relaxação linear associada. E portanto, de maneira geral, uma partição de um subproblema P_j é dada por $P_j = P_{j+1} \cup P_{j+2}$ onde:

$$P_{j+1} = P_j \cap \{x \in \mathcal{R}_+^n / x_i \leq \lfloor \bar{x}_i \rfloor\}$$

e

$$P_{j+2} = P_j \cap \{x \in \mathcal{R}_+^n / x_i \geq \lfloor \bar{x}_i \rfloor + 1\}.$$

Outras escolhas para o par (d, d_0) são sugeridas em [NW – 1988].



Critérios de Poda

Suponha que a relaxação P_j do nó j da árvore de enumeração tenha uma solução ótima $\bar{x}^j$. Não haverá necessidade de particionarmos este nó, isto é, *podamos* este nó quando um dos critérios abaixo for verificado:

- i. O problema P_j é inviável;
- ii. A solução $\bar{x}^j$ de P_j é viável para (PI) ou
- iii. $\underline{c}_j \geq \bar{c}$, onde $\bar{c}$ é o valor de alguma solução viável para (PI).

Seleção do Nó

Conforme a partição vai sendo criada, os subproblemas gerados vão sendo armazenados em uma lista. Desse modo, dada uma lista de espera de subproblemas (que denotaremos por L) chamados de nós ativos, devemos decidir que nó será examinado na próxima iteração do algoritmo BB. Nemhauser e Wolsey [NW - 1988] fazem algumas sugestões.

Uma regra comum para selecionar um subproblema da lista de espera é conhecido como “*depth first search plus backtracking*”, ou LIFO (*last in first out* - o último a entrar, o primeiro a sair). Se o nó corrente (nó pai) é ramificado, o próximo a ser considerado é uma de suas ramificações (ou nó filho). *Backtracking* acontece quando um nó é podado e voltamos no caminho do nó corrente em direção ao nó raiz até encontrar o primeiro nó (se houver) que tenha uma ramificação que não tenha sido considerada. Esta regra tem duas vantagens principais:

- i. A relaxação corrente do nó filho é obtida da relaxação linear do nó pai, pela adição de uma restrição de limite inferior ou superior para as variáveis;
- ii. A segunda vantagem é que esta regra ajuda a encontrar soluções viáveis, pois é mais provável que elas estejam em um nível *mais profundo* da árvore de enumeração. É claro que esta vantagem desaparece assim que uma solução inteira é encontrada.



Uma estratégia diferente é “*breadth first search*” onde todos os nós em um dado nível são examinados antes que qualquer nó do próximo nível seja considerado. O nível de um nó na árvore de enumeração é o número de nós no único caminho entre ele e a raiz da árvore. Esta regra tende a gerar poucos nós, embora não seja muito prática quando se usa a relaxação linear.

Seleção da Variável a ser Ramificada

Um outro critério importante a ser considerado no algoritmo BB é o de decidir que variável será selecionada para a criação de subproblemas, dado que a relaxação linear poderá fornecer mais de uma variável com valor fracionário. Nemhauser e Wolsey [NW - 1988] citam os critérios abaixo.

Dado um nó i e a solução associada, $\bar{x}^i$, a escolha da variável está limitada ao conjunto de índices $N^i = \{j \in N / \bar{x}_j^i \text{ é não inteira}\}$. Evidências empíricas têm mostrado que a escolha de um $j \in N^i$ pode ter um resultado forte no tempo gasto pelo algoritmo. Frequentemente há poucas variáveis que uma vez fixadas em valores inteiros retornam outras variáveis inteiras na solução linear resultante. Contudo, não há uma maneira robusta para identificar estas variáveis. Uma maneira comum é escolher a variável de acordo com uma dada prioridade. Quando nenhuma prioridade é estabelecida ou há mais que uma variável fracionária com a maior prioridade, outros critérios têm de ser usados. Outros critérios incluem a escolha da variável com a menor (ou maior) parte fracionária, ou ainda a escolha da variável baseado no julgamento de quanto o limite inferior do nó i aumentará quando esta variável se tornar inteira.

Após esta discussão, já podemos elaborar um algoritmo. Denotaremos por L a lista de espera de nós. O algoritmo é finalizado quando a lista de nós está vazia. O algoritmo esta baseado em [BM - 1992], [Rangel - 1995] e [NW - 1988].

Algoritmo Branch and Bound

Passo 1: (Inicialização). Seja $L = \{PL\}$, $\underline{c}_j = -\infty$ e $\bar{c} = \infty$.



Passo 2: (*Seleção do nó*). Se $L = \emptyset$, **pare**, vá ao passo 8. Caso contrário, selecione um subproblema da lista L e remova-o.

Passo 3: (*Resolva a relaxação corrente*). Resolva o subproblema

$$\begin{array}{ll} \text{Min.} & c^T x \\ \text{S. a} & x \in P_j \end{array}$$

Passo 4: (*Podando por inviabilidade*). Se o problema corrente é inviável, o nó é podado e vá ao passo 2. Caso contrário, denote por $\underline{c}_j = c^T \bar{x}^j$ o valor da solução ótima do subproblema.

Passo 5: (*Podando por integralidade*). Se $\bar{x}^j$ é inteiro, o nó é podado. Faça $\bar{c} = \min\{\bar{c}, \underline{c}_j\}$. Volte ao passo 2.

Passo 6: (*Podando pelo limite*). Se $\underline{c}_j \geq \bar{c}$, pode este nó e volte ao passo 2.

Passo 7: (*Ramificação*). Escolha uma variável fracionária $\bar{x}_i^j$ e particione P_j . Adicione os subproblemas à lista L e vá ao passo 2.

Passo 8: (*Término*). Se $\bar{c} = \infty$, o problema (PI) não tem solução. Se $\bar{c} < \infty$, a solução associada é ótima.

4.2 O Método Branch and Cut

O *Método Branch and Cut* (BC) se diferencia do método clássico BB por usar em cada nó da árvore BB o método de Planos de Cortes, ou seja, o método BC adiciona inequações válidas no problema a fim de eliminar soluções fracionárias durante a fase BB do algoritmo e assim melhorar a qualidade dos limitantes em cada nó.

A eficiência do método Branch and Bound depende de quão próximo está a sua relaxação (PL) de (PI) e como encontrar uma solução viável rapidamente. Se a relaxação do (PL) é ruim, um número grande de nós serão resolvidos independentemente da estratégia usada para seleção de subproblemas e variável. Por outro lado, se uma boa relaxação para



(PL) é obtida, pode ser que o problema seja resolvido no nó raiz. O método de Planos de Cortes fornece uma boa alternativa para melhorar a relaxação (PL).

A técnica BC foi proposta por Crowder, Johnson e Padberg (1983), embora eles tenham usado os cortes somente no nó raiz da árvore BB. O termo BC foi estabelecido por Padberg e Rinaldi (1987, 1993) que generalizou a idéia e a aplicou com sucesso ao problema do caixeiro viajante de larga escala [Rangel - 1995].

O algoritmo BC executa todos os passos do algoritmo BB, sem exceção, e incorpora um novo passo fundamental: a geração dos planos de cortes. Assim, além da seleção de subproblemas e seleção de variável, surge um terceiro parâmetro a ser incorporado ao método, que é a geração ou não de planos de corte.

Quando há a necessidade de gerar cortes, é razoável verificar se há mudanças significativas no valor da função objetivo. Se nas últimas k iterações (por exemplo, $k = 10$) não houve uma melhora na relaxação linear corrente, de pelo menos $p\%$ (por exemplo, $p = 0.01$), a geração de planos de cortes é interrompida e o subproblema é ramificado. Este processo é conhecido como “*Tailing-off*” [PR - 1991] e [Thienel - 1995].

A fase de *separação* (ou geração de cortes) é a parte central do algoritmo BC, pois nesta fase tentamos encontrar inequações válidas para adicioná-las ao (PL). Dizemos que estas desigualdades são válidas globalmente se são válidas para cada subproblema da árvore BC. Caso contrário são válidas localmente.

A seguir descreveremos um algoritmo BC, baseado em [BM- 1992], [Rangel - 1995] e [NW - 1988].

Algoritmo Branch and Cut

Passo 1: (Inicialização). Seja $L = \{PL\}$, $\underline{c}_j = -\infty$ e $\bar{c} = \infty$.

Passo 2: (Seleção do nó). Se $L = \emptyset$, pare, vá ao passo 10. Caso contrário, selecione um subproblema da lista L e remova-o.

Passo 3: (Resolva a relaxação corrente). Resolva o subproblema



$$\text{Min. } c^T x$$

$$\text{S. a } x \in P_j$$

Passo 4: (*Podando por inviabilidade*). Se o problema corrente é inviável, o nó é podado e vá ao passo 2. Caso contrário, denote por $\underline{c}_j = c^T \bar{x}^j$ o valor da solução ótima do subproblema.

Passo 5: (*Podando por integralidade*). Se $\bar{x}^j$ é inteiro, o nó é podado. Faça $\bar{c} = \min\{\bar{c}, \underline{c}_j\}$. Volte ao passo 2.

Passo 6: (*Podando pelo limite*). Se $\underline{c}_j \geq \bar{c}$, pode este nó e volte ao passo 2.

Passo 7: (*Tailing-off*). Se há continuidade na geração de cortes, vá ao passo 8. Caso contrário, vá ao passo 9.

Passo 8: (*Geração de planos de cortes*). Gere inequações globalmente válidas para P^j que sejam violadas por $\bar{x}^j$. Adicione os cortes a P_j e volte ao passo 3.

Passo 9: (*Ramificação*). Escolha uma variável fracionária $\bar{x}_i^j$ e particione P_j . Adicione os problemas à lista L e vá ao passo 2.

Passo 10: (*Término*). Se $\bar{c} = \infty$, o problema (PI) não tem solução. Se $\bar{c} < \infty$, a solução associada é ótima.

Até aqui estivemos preocupados em revisar brevemente a estrutura básica dos métodos BB e BC, sem mencionar qual método será usado para resolver a relaxação de (PI). Na próxima seção discutiremos quais mudanças devem ser feitas para o uso de método de Pontos Interiores na resolução dos subproblemas.

4.3 Uso de Pontos Interiores no Método Branch and Cut

Abordaremos aqui alguns detalhes importantes do método BC quando usamos o método de Pontos Interiores na resolução dos subproblemas. A eficiência do método de Pontos Interiores neste caso depende da eficiência de sua implementação e também dos critérios usados para decidir quando paramos a resolução da relaxação corrente e de como



aproveitamos a solução obtida como um *ponto de partida* para o próximo subproblema. Uma heurística primal também pode ser usada para verificar quando uma variável está tendendo a um valor fracionário.

Como usaremos planos de cortes apenas no nó raiz o processo de reiniciação do algoritmo e de terminação antecipada estão descritos no capítulo 3. Apresentaremos aqui apenas as modificações necessárias para aplicá-las ao método de Pontos Interiores na resolução dos subproblemas associados à árvore BB. Ressaltamos também que a discussão abaixo baseia-se em problemas de (PI) zero-um.

Ponto de Partida no Nó da Árvore

O método usado para reiniciar o algoritmo no nó *j* da árvore BB depende do algoritmo de Pontos Interiores usado. Em Mitchell [Mitchell - 1995] o autor sugere o método de Barreira Primal-Dual com os aspectos descritos a seguir.

Suponha que o subproblema gerado foi criado por fixar uma variável com valor fracionário no seu limite superior ou inferior, em particular, por uma variável *x*₀ no problema corrente, ou (PL *pai*):

(PL <i>pai</i>)	Min. $c^T x + c_0 x_0$	Problema no formato padrão	Min. $c^T x + c_0 x_0$
	S. a $Ax + a_0 x_0 = b$		S. a $Ax + a_0 x_0 = b$
	$x \leq e$		$x + s = e$
	$x_0 \leq 1$		$x_0 + s_0 = 1$
	$x, x_0 \geq 0,$		$x, x_0 \geq 0,$

onde $A \in \mathbb{Z}^{m \times n}$, $a_0, b \in \mathbb{Z}^m$, $x, c \in \mathbb{Z}^n$, c_0, x_0 são escalares e *s*, *s*₀ são variáveis de folga primais e $e = (1, 1, \dots, 1)^T$. O problema gerado, ou (PL *filho*), tem a forma:

(PL <i>filho</i>)	Min. $c^T x$	Problema no formato padrão	Min. $c^T x$
	S. a $Ax = \bar{b}$		S. a $Ax = \bar{b}$
	$x \leq e$		$x + s = e$
	$x \geq 0,$		$x \geq 0,$

onde $\bar{b} = b - a_0$ quando x_0 é fixo em 1. Uma solução aproximada $x = x^*$, $x_0 = x_0^*$ para o (PL *pai*) é conhecida. Como criamos este subproblema (PL *filho*) particular, x_0^* é fracionário, e portanto, x^* é inviável para o (PL *filho*). Se examinarmos o problema dual corrente, ou (PD *pai*):

$$\begin{aligned}
 & \text{Max. } b^T y - e^T w - w_0 \\
 \text{(PD } \textit{pai}) \quad & \text{S. a } A^T y + z - w = c \\
 & a_0^T y + z_0 - w_0 = c_0 \\
 & z, z_0, w, w_0 \geq 0 \\
 & y \text{ irrestrito,}
 \end{aligned}$$

e

$$\begin{aligned}
 & \text{Max. } \bar{b}^T y - e^T w \\
 \text{(PD } \textit{filho}) \quad & \text{S. a } A^T y + z - w = c \\
 & z, w \geq 0 \\
 & y \text{ irrestrito,}
 \end{aligned}$$

notamos que a solução aproximada $y = y^*$, $z = z^*$, $w = w^*$ e $w_0 = w_0^*$ para o (PD *pai*) é viável também para o (PD *filho*). Portanto, é possível reiniciar o algoritmo usando o método de Pontos Interiores Inviável. É possível que alguns componentes de x^* estejam próximos do valor de seus limites, e estes componentes podem ser modificados para pontos mais interiores, ou seja, as componentes pequenas serão aumentadas e as grandes diminuídas. Similarmente, se alguns componentes das variáveis y^* , z^* e w^* ou as variáveis de folga primal ou dual são menores que alguma tolerância, eles podem ser aumentados para a tolerância dada. Assim, podemos iniciar o método de Pontos Interiores no problema *filho* se temos armazenados x^* , y^* , z^* e w^* . É útil usar um primeiro passo de centralização puro antes de atualizar a métrica dual μ de maneira apropriada.

Também pode-se iniciar a solução do problema *filho* de uma solução do (PL *pai*) que foi encontrada antes de x^* . Esta solução antecipada será mais distante da otimalidade para (PL *pai*) quanto para x^* , mas ela será uma solução inicial melhor para o (PL *filho*) pois é mais centralizada, com os componentes não-básicos existentes relativamente grandes.



Experimentos preliminares mostram que esta aproximação é promissora, mas necessita de maiores investigações [Mitchell - 1995].

Terminando a Relaxação Corrente antecipadamente

A solução do subproblema corrente pode ser usada como um *ponto de partida* para o próximo subproblema, mas para este propósito o método de Pontos Interiores deve ter pontos *centrados*, ou seja, bem localizados no interior da região viável e, portanto, pode ser útil terminarmos a relaxação corrente antecipadamente.

Porém há uma desvantagem ao pararmos a relaxação corrente antecipadamente: o subproblema corrente poderá ser dividido em dois subproblemas, quando ele poderia ter sido podado se tivéssemos resolvido até a otimalidade. Portanto, podemos incluir alguma proteção para evitar esta situação, como por exemplo, usar o valor dos limites inferior e superior que um subproblema fornece através da solução corrente primal dual. Este critério também pode ser usado para regular a decisão de ramificação.

Mitchell [Mitchell - 1995] sugere três testes para prevenir a ramificação antecipada: (i) a solução dual tem de ser viável, (ii) a inviabilidade primal relativa tem de estar em torno de 10%, e (iii) o valor da função objetivo dual não pode aumentar rapidamente de iteração a iteração, pois é provável que o subproblema seja sondado pelo limite dentro de uma ou duas iterações.

A viabilidade dual pode ser mantida na árvore BB de modo que o primeiro critério é basicamente técnico. Em toda ramificação uma variável é fixada e a inviabilidade primal pode ser recuperada em poucas iterações. Portanto, note que se a iteração primal corrente é viável e se seu valor objetivo é maior do que a do (PI) corrente, então este critério evita a ramificação caso a diferença entre as soluções dual e primal correntes tenha sido reduzida em pelo menos metade na última iteração.

Heurística Primal

Uma das maneiras de se acelerar um algoritmo do tipo BC (ou BB) é a utilização de uma heurística primal. As heurísticas servem, por exemplo, para determinar se alguma das



variáveis está convergindo para um valor fracionário e assim ramificar o problema antecipadamente.

Em [BM - 1992], os autores usaram uma heurística deste tipo quando trabalhando com método de Pontos Interiores Primal-Dual para variáveis canalizadas, em específico, para variáveis 0-1. Os autores baseiam-se no fato de que as razões x_i^{k+1}/x_i^k e s_i^{k+1}/s_i^k tendem a 1 e w_i^{k+1}/w_i^k e z_i^{k+1}/z_i^k tendem a zero conforme a solução aproxima-se da otimalidade. Então, eles definem que uma variável 0-1 é fracionária quando:

$$\left| \frac{x_i^{k+1}}{x_i^k} - 1 \right| < 0.1$$

$$\left| \frac{s_i^{k+1}}{s_i^k} - 1 \right| < 0.1 \quad (4.1)$$

$$\frac{w_i^{k+1}}{w_i^k} < 0.6$$

$$\frac{z_i^{k+1}}{z_i^k} < 0.6$$

Assim, usando o critério (4.1) podemos parar o algoritmo antecipadamente e seguir para a geração de planos de cortes. Observe que s é a variável de folga primal.

Após toda essa discussão podemos propor um algoritmo BC supondo o uso do método de Pontos Interiores para resolver as relaxações lineares. O algoritmo foi elaborado a partir de [BM - 1992], [Rangel - 1995] e [NW - 1988].

Algoritmo Branch and Cut

Passo 1: (Inicialização). Seja $L = \{PL\}$, $\underline{c}_j = -\infty$ e $\bar{c} = \infty$, encontre $(x, y, z) \in \Omega^0$ e defina τ_1 (otimalidade).



Passo 2: (*Resolva a relaxação linear*). Remova o (PL) da lista e faça uma iteração do método de Pontos Interiores.

(a) Se o problema é inviável, vá ao passo 8.

(b) Se $gap < \tau_1$, seja $\bar{x}^0$ a sua solução. Se $\bar{x}^0$ é inteiro, faça $c_0 = c^T \bar{x}^0$ e vá ao passo 8. Caso contrário, vá ao passo 7.

(c) (*Tailing-off*). Se há continuidade na geração de cortes, vá ao passo 3. Caso contrário, vá ao passo 7.

Passo 3: (*Geração de Planos de Cortes*). Gere inequações válidas para P^l que seja violadas pela solução da relaxação linear corrente $\bar{x}^j$. Adicione os cortes a P e volte ao passo 2.

Passo 4: (*Seleção do nó*). Se $L = \emptyset$, **pare**, vá ao passo 8. Caso contrário, selecione um subproblema da lista L e remova-o. Vá ao passo 5.

Passo 5: (*Resolva o subproblema corrente*). Faça uma iteração do método de Pontos Interiores.

Passo 6: (*Poda*).

(a) (*Inviabilidade*). Se o problema corrente é inviável, o nó é podado. Vá ao passo 4. Caso contrário, denote por $\underline{c}_j = c^T \bar{x}^j$ o valor da solução do subproblema.

(b) (*Heurística Primal*). Use uma heurística para verificar se P_j está convergindo para um valor fracionário. Se estiver, vá ao passo 7.

(c) (*Integralidade*). Se $\bar{x}^j$ é inteiro, o nó é podado. Faça $\bar{c} = \min\{\bar{c}, \underline{c}_j\}$, e volte ao passo 4.

(d) (*Limite*). Se $\underline{c}_j \geq \bar{c}$, pode este nó e volte ao passo 4.

(e) Caso contrário, volte ao passo 5.



Passo 7: (Ramificação). Escolha uma variável fracionária $\bar{x}_i^j$ e particione P_j . Adicione os problemas à lista L e vá ao passo 4.

Passo 8: (Término): Se $\bar{c} = \infty$, o problema (PI) não tem solução. Se $\bar{c} < \infty$, a solução associada é ótima.

No capítulo a seguir, veremos uma aplicação do uso de Pontos Interiores no método de Planos de Cortes para a resolução do Problema de Ordenamento Linear.

Neste capítulo apresentaremos uma aplicação do uso do Método de Planos de Cortes com Pontos Interiores para resolver o problema de Ordenamento Linear (Linear Ordering) [MIR - 1996] e [GJR - 1984].

O Problema de Ordenamento Linear é um problema de otimização combinatorial com uma grande variedade de aplicações, como por exemplo:

Agregação de preferências: Suponha um conjunto de n investimentos, que serão oferecidos a um determinado grupo de pessoas. As pessoas podem ordenar estes investimentos de acordo com graus de preferência. Quando $i < j$, significa de vez que esta pessoa prefere o investimento i ao investimento j . Portanto, para esta informação em um problema de ordenamento, cada solução fornece uma seqüência de "ordenação consistente", ou seja, a ordenação que mais se aproxima das decisões da clientela.

Datação arqueológica: Uma ordem cronológica mais provável pode ser encontrada para um conjunto de cerâmicas, por exemplo, se a elas forem atribuídas pesos representativos de se achar uma peça do tipo i em uma escavação mais profunda do que uma peça do tipo j .

O problema de Ordenamento Linear é NP-Completo e ainda não foi obtida uma descrição completa do seu envoltório convexo [MIR - 1996].

5.1 Formulação do Problema

Antes de apresentarmos a formulação do problema de Ordenamento Linear vejamos algumas definições.

Capítulo

5

O Problema de Ordenação Linear

Neste capítulo apresentaremos uma aplicação do uso do Método de Planos de Cortes com Pontos Interiores para resolver o problema de Ordenação Linear (*Linear Ordering*) [MB - 1996] e [GJR - 1984].

O *Problema de Ordenação Linear* é um problema de otimização combinatória com uma grande variedade de aplicações, como por exemplo:

Agregação de preferências: Suponha um conjunto de n investimentos, que serão oferecidos a um determinado grupo de pessoas. As pessoas podem ordenar estes investimentos de acordo com graus de preferência. Contando-se o número de vezes que cada pessoa preferiu o investimento i ao investimento j podemos juntar esta informação em um problema de ordenação, cuja solução fornece uma espécie de “ordenação consistente”, ou seja, a ordenação que mais se aproxima dos desejos da clientela.

Datação arqueológica: Uma ordem cronológica mais provável pode ser encontrada para um conjunto de cerâmicas, por exemplo: se a elas forem atribuídos pesos representativos de se achar uma peça do tipo i em uma escavação mais profunda do que uma peça do tipo j .

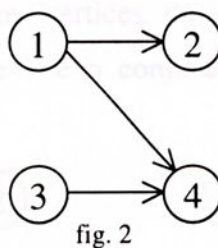
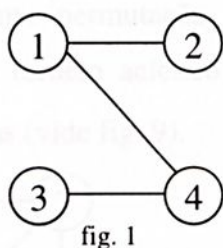
O problema de Ordenação Linear é NP-Completo e ainda não foi obtida uma descrição completa de seu envoltório convexo [MB - 1996].

5.1 Formulação do Problema

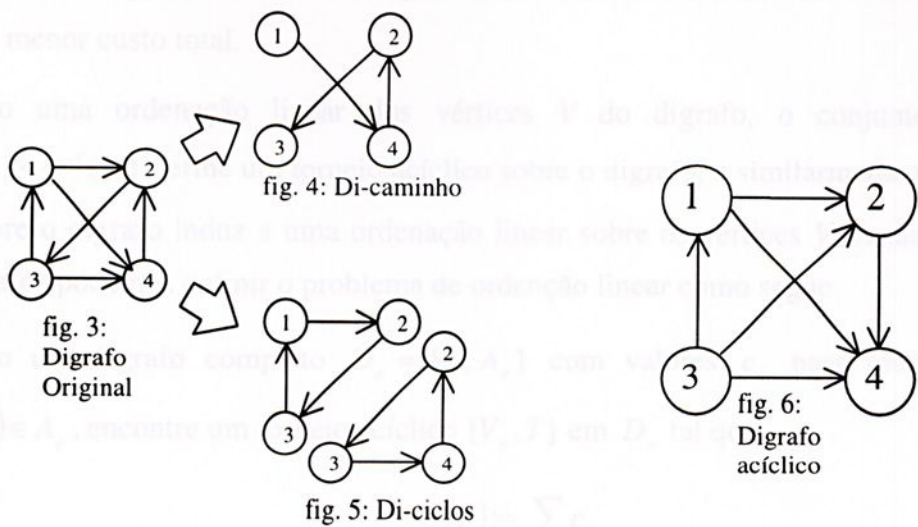
Antes de apresentarmos a formulação do problema de Ordenação Linear vejamos algumas definições.



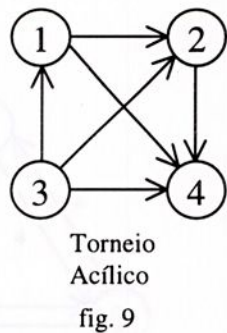
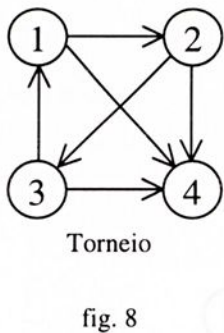
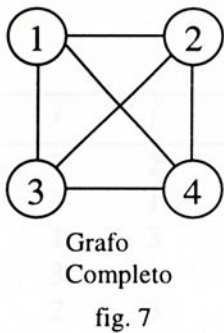
- **Grafo:** um grafo $G = [V, E]$ consiste de um conjunto finito e não vazio de *vértices* V juntamente com um conjunto finito de *arestas* E , onde cada elemento $e \in E$ é definido como um par não-ordenado de vértices i e j em V , $e = ij$. Suporemos que G é um grafo simples, isto é, não contém arestas do tipo $e = ii$. E mais, dizemos que um grafo é *completo* se existe uma aresta para cada par de vértices ij (vide fig. 7). Um grafo pode ser representado graficamente através de um diagrama onde os círculos numerados representam os vértices e um segmento de reta entre dois vértices representa uma aresta (vide fig. 1).
- **Digrafo:** um digrafo $D = [V, A]$ consiste de um conjunto não-vazio de vértices V juntamente com um conjunto finito de *arcos* A que são pares ordenados de V . Se $G = [V, E]$ é um grafo, um digrafo $D = [V, A]$ é chamado de uma orientação de G se, para todo $ij \in E$, A contém arcos (i, j) ou arcos (j, i) , mas não ambos. Um digrafo é *completo* se existe um arco (i, j) e (j, i) para cada par de vértices ij . Para representarmos graficamente um digrafo, acrescentamos uma flecha indicando a orientação do arco (vide fig. 2).



- **Di-caminho:** um conjunto de arcos $P = \{(v_1, v_2), (v_2, v_3), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k)\}$ em $D = [V, A]$ com $v_i \neq v_j$ para todo $i \neq j$ é chamado de um (v_1, v_k) -*di-caminho* de tamanho $k-1$ (vide fig. 4).
- **Diciclo:** Se P é um (v_1, v_k) -*di-caminho* e $(v_k, v_1) \in A$, $C = P \cup \{(v_k, v_1)\}$ é um *diciclo* de tamanho k ou k -*diciclo* (vide fig. 5).
- **Digrafo acíclico:** um digrafo $D = [V, A]$ que não contém nenhum diciclo é chamado de *acíclico* (vide fig. 6).



- *Torneio*: um torneio é um digrafo $D=[V, A]$ tal que para todo par de vértices i e j , o conjunto A contém exatamente um dos dois arcos (i, j) ou (j, i) . Assim, um torneio é uma orientação do grafo completo, que denotaremos por K_p (vide fig. 8).
- *Torneio Acíclico*: é um torneio que não contém diciclos. Observe que todo torneio acíclico corresponde a uma permutação ou ordenação dos vértices do grafo, e vice-versa. Denotaremos um torneio acíclico por $[V, T]$, onde V é o conjunto de vértices e T o conjunto de arestas (vide fig. 9).



Uma *ordenação linear* de um conjunto finito V com $|V|=p$ é uma função bijetora $\sigma:\{1,2,...,p\}\rightarrow V$. Se $i, j\in V$ e $\sigma^{-1}(i) < \sigma^{-1}(j)$, dizemos que i está antes de j . Para todo par de vértices $i, j\in V$, associamos valores c_{ij} e c_{ji} que correspondem ao custo de se ter i antes de j e j antes de i , respectivamente. Então o valor total de uma ordenação linear é dado por:

$$\sum_{\sigma^{-1}(i) < \sigma^{-1}(j)} c_{ij}.$$

A solução do problema de Ordenação Linear consiste em encontrar o *torneio acíclico* que tenha o menor custo total.

Dado uma ordenação linear dos vértices V do digrafo, o conjunto de arcos $\{(i, j): \sigma^{-1}(i) < \sigma^{-1}(j)\}$ define um torneio acíclico sobre o digrafo, e similarmente todo torneio acíclico sobre o digrafo induz a uma ordenação linear sobre os vértices V . Assim, usando a teoria de grafos podemos definir o problema de ordenção linear como segue.

Dado um digrafo completo $D_p = [V_p, A_p]$ com valores c_{ij} para todos os arcos $(i, j) \in A_p$, encontre um torneio acíclico $[V_p, T]$ em D_p tal que:

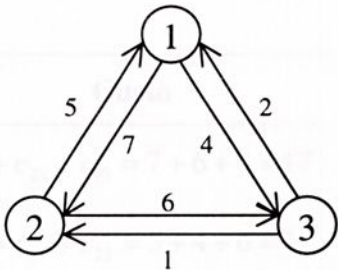
$$c(T) := \sum_{(i,j) \in T} c_{ij}$$

é minimizado.

Para ajudar na compreensão dos conceitos aqui envolvidos, faremos manualmente um problema envolvendo apenas três nós. Apesar de pequeno, cremos que este exemplo será esclarecedor.

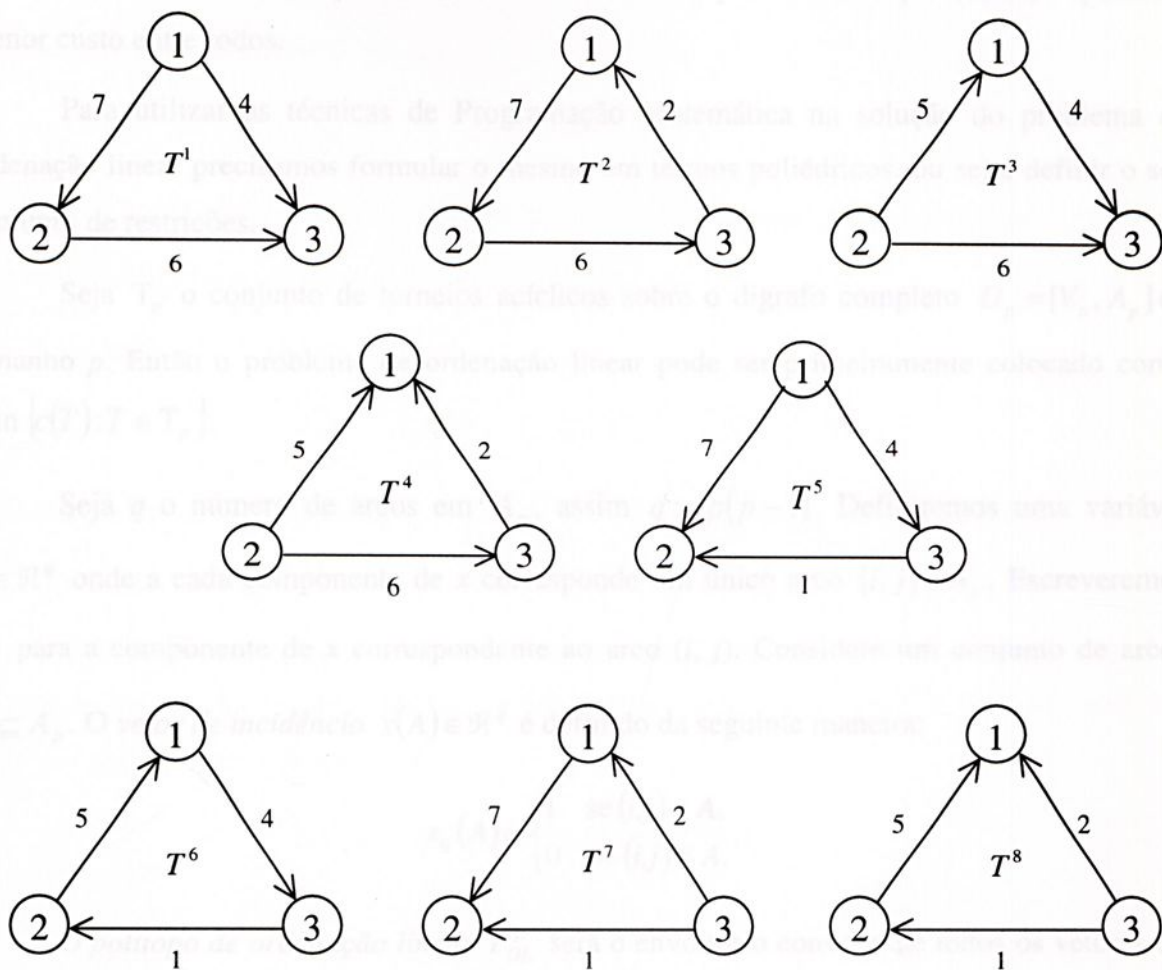
Seja então o digrafo completo $D_p = [V_p, A_p]$, com conjunto de vértices $V_p = \{1,2,3\}$ e arcos $A_p = \{(1,2), (2,1), (1,3), (3,1), (2,3), (3,2)\}$. Os custos associados a termos o vértice i antes de j estão resumidos na tabela e digrafo abaixo.

i	j	c_{ij}
1	2	7
2	1	5
1	3	4
3	1	2
2	3	6
3	2	1



O que temos de fazer agora é, com base no digrafo completo acima, gerar todos os torneios (ou seja, combinar de todas maneiras possíveis os 6 arcos), excluir os cíclicos, calcular o valor de cada ordenação resultante e finalmente escolher o de menor valor.

Exibamos primeiramente todos os torneios:



Descartamos agora os torneios T^2 e T^6 , pois são cíclicos. Dos torneios restantes, extraímos as ordenações, resumidas na tabela abaixo:

Torneio	Ordenação	Custo
T^1	(1,2,3)	$c_{12} + c_{23} + c_{13} = 7 + 6 + 4 = 17$
T^3	(2,1,3)	$c_{21} + c_{13} + c_{23} = 5 + 4 + 6 = 15$
T^4	(2,3,1)	$c_{23} + c_{31} + c_{21} = 6 + 2 + 5 = 13$
T^5	(1,3,2)	$c_{13} + c_{32} + c_{12} = 4 + 1 + 7 = 12$
T^7	(3,1,2)	$c_{31} + c_{12} + c_{32} = 2 + 7 + 1 = 10$
T^8	(3,2,1)	$c_{32} + c_{21} + c_{31} = 1 + 5 + 2 = 8$

Enfim, concluímos que o torneio T^8 é o ótimo, pois a ordenação (3,2,1) – possui o menor custo entre todos.

Para utilizar as técnicas de Programação Matemática na solução do problema de ordenação linear precisamos formular o mesmo em termos poliédricos, ou seja, definir o seu conjunto de restrições.

Seja T_p o conjunto de torneios acíclicos sobre o digrafo completo $D_p = [V_p, A_p]$ de tamanho p . Então o problema de ordenação linear pode ser primeiramente colocado como $\min \{c(T) : T \in T_p\}$.

Seja q o número de arcos em A_p , assim $q := p(p-1)$. Definiremos uma variável $x \in \mathbb{R}^q$ onde a cada componente de x corresponde um único arco $(i, j) \in A_p$. Escreveremos x_{ij} para a componente de x correspondente ao arco (i, j) . Considere um conjunto de arcos $A \subseteq A_p$. O vetor de incidência $x(A) \in \mathbb{R}^q$ é definido da seguinte maneira:

$$x_{ij}(A) = \begin{cases} 1 & \text{se } (i, j) \in A, \\ 0 & \text{se } (i, j) \notin A. \end{cases}$$

O politopo de ordenação linear P_{OL}^p será o envoltório convexo de todos os vetores do conjunto de arcos dos torneios acíclicos em D_p . Assim,

$$P_{OL}^p = \text{conv} \{x(T) \in \mathbb{R}^q : T \in T_p\}. \quad (5.1)$$

Todo torneio acíclico corresponde a um vértice de P_{OL}^p , e vice-versa. Portanto, resolvendo o (PL) $\min \{c^T x, x \in P_{OL}^p\}$, resolvemos o problema de Ordenação Linear. Contudo, perceba que (PL) não está numa forma conveniente para nenhum dos métodos clássicos de resolução (Simplex ou Pontos Interiores), pois não temos P_{OL}^p definido em termos de inequações, mas sim em termos de seus pontos extremos.

Infelizmente ainda não é conhecida uma caracterização satisfatória de P_{OL}^p em termos de suas facetas [MB – 1996]. Porém, aproximações razoáveis de P_{OL}^p já são conhecidas, como afirmam [GJR – 1984] e [MB – 1996]. Vejamos algumas destas aproximações.

Seja $D_p = [V_p, A_p]$ o digrafo completo de ordem p , com $p \geq 3$. Então a solução do sistema de equações abaixo:

$$x_{ij} + x_{ji} = 1, \quad \forall i, j \in V_p, \quad i \neq j \quad (5.2)$$

é igual ao envoltório afim de P_{OL}^p . O que estas inequações estão expressando na verdade é o fato de só admitirmos um arco de cada vez, (i, j) ou (j, i) .

As inequações triviais

$$x_{ij} \geq 0 \quad \text{e} \quad x_{ij} \leq 1 \quad (5.3)$$

definem facetas de P_{OL}^p para todo $(i, j) \in A_p$. Cada inequação $x_{ij} \geq 0$ é equivalente a $x_{ji} \leq 1$ (e vice-versa) e a nenhuma outra inequação trivial.

O problema de Ordenação Linear está associado a torneios acíclicos. Note que o fato do torneio ser acíclico implica que não podem haver 3-diciclos no digrafo. Este fato pode ser expresso pelo conjunto de inequações:

$$e^T x(C) \leq 2 \quad \text{para todo 3-diciclo } C \in A_p, \quad (5.4)$$

ou seja:

$$x_{ij} + x_{jk} + x_{ki} \leq 2, \quad \text{para todo 3-diciclo } C \in A_p, \quad i \neq j \neq k,$$

que também definem facetas em P_{OL}^p . Ainda mais: nenhuma dessas inequações é equivalente às constantes de (5.3). As inequações (5.4) são chamadas *inequações triangulares*.

Definamos então os novos politopos:

$$P_1^p = \{x \in \mathbb{R}^q : x \text{ satisfaz (5.2) e (5.3)}\} \quad (5.5)$$

$$P_2^p = \{x \in \mathbb{R}^q : x \text{ satisfaz (5.2), (5.3) e (5.4)}\}. \quad (5.6)$$

Então, $P_{OL}^p \subseteq P_2^p \subseteq P_1^p$. Os politopos P_1^p e P_2^p são na verdade aproximações para P_{OL}^p .

Pode ainda ser mostrado que [MB – 1996]:

$$P_{OL}^p = \text{conv}\{x \in P_2^p : x \text{ inteiro}\}.$$

Dessa maneira, se a solução ótima do problema $\min\{c^T x : x \in P_2^p\}$ é inteira, então é também solução ótima do problema de Ordenação Linear $\min\{c^T x : x \in P_{OL}^p\}$. Assim, temos que:

$$\min\{c^T x : x \in P_2^p\} = \min\{c^T x : x \in P_{OL}^p\},$$

e podemos dizer que P_2^p é uma boa aproximação para o politopo do problema de Ordenação Linear.

Finalmente, faremos as últimas considerações que nos levarão a uma forma mais “tratável” do problema de Ordenação Linear.

Perceba que da definição de torneio, não podemos ter ao mesmo tempo os arcos (i, j) e (j, i) . Portanto, se $x_{ij} = 0$, necessariamente $x_{ji} = 1$ para todo $ij \in V_p$. Desta forma, podemos eliminar metade das variáveis. Este fato também se justifica pelas equações (5.2), se rescritas:

$$x_{ji} = 1 - x_{ij}, \quad 1 \leq i < j \leq p.$$

Podemos então executar esta substituição para todo x_{ji} tal que $j < i$ em todas as inequações do problema. Desta forma diminuimos o número de variáveis para $\bar{q} = \binom{p}{2} = \frac{p(p-1)}{2}$.

Esta substituição, além de reduzir o número de variáveis do problema para a metade, define também uma projeção de $\mathcal{R}^q$ em $\mathcal{R}^{\bar{q}}$. Desta forma, dado um politopo $P^p \subseteq \mathcal{R}^q$ denotaremos sua projeção em $\mathcal{R}^{\bar{q}}$ por $\bar{P}^p$. Através da projeção definida acima chegamos a equivalência: dado um problema linear $\min\{c^T x : x \in P^p, P^p \subseteq \text{envoltório afim de } P_{OL}^p\}$, existe um problema linear equivalente $\min\{\bar{c}^T x : x \in \bar{P}^p\}$ com $\bar{c}_{ij} = c_{ij} - c_{ji}$, $1 \leq i < j \leq p$.

Esta projeção em especial é facilmente inversível, bastando que se acrescente a $\mathcal{R}^{\bar{q}}$ as variáveis que foram retiradas na projeção. Concluimos então que esta projeção é na verdade uma bijeção, e que o valor de um ponto em P^p difere do valor de sua respectiva projeção em $\bar{P}^p$ pela constante $\sum_{i>j} c_{ij}$.

Com base na discussão prévia, vamos obter a projeção de P_{OL}^p . As inequações em (5.3) podem ser rescritas como

$$0 \leq x_{ij} \leq 1, \quad 1 \leq i < j \leq p. \quad (5.7)$$

As inequações triangulares correspondentes aos 3-diciclos serão rescritas com base em um 3-diciclo geral: para cada tripla $\{i, j, k\} \in V_p$, com $1 \leq i < j < k \leq p$, há duas facetas de P_{OL}^p definidas por

$$x_{ij} + x_{jk} + x_{ki} \leq 2$$

$$x_{ji} + x_{kj} + x_{ik} \leq 2.$$

Usando-se a projeção acima definida, obtemos as duas facetas equivalentes em $\bar{P}_{OL}^p$, correspondentes aos diciclos $\{i, j, k\}$ e $\{k, j, i\}$:

$$x_{ij} + x_{jk} - x_{ik} \leq 1 \quad (5.8)$$

$$-x_{ij} - x_{jk} + x_{ik} \leq 0. \quad (5.9)$$

Definimos de forma análoga os politopos $\bar{P}_1^p$ – satisfazendo (5.7) – e $\bar{P}_2^p$ – satisfazendo (5.7) a (5.9), ou seja:

$$\bar{P}_1^p = \{x \in \mathbb{R}^q : x \text{ satisfaz (5.7)}\},$$

$$\bar{P}_2^p = \{x \in \mathbb{R}^q : x \text{ satisfaz (5.7), (5.8) e (5.9)}\}.$$

Assim, no presente caso, para resolvermos o problema de Ordenação Linear, resolvemos o seguinte problema linear $\min\{c^T x : x \in \bar{P}_2^p\}$, compreendendo $\binom{p}{2}$ variáveis e $2\binom{p}{2} + 2\binom{p}{3}$ restrições.

Voltando ao exemplo com três vértices abordado acima, seu problema linear equivalente tomaria a forma:

$$\text{Min. } 7x_{12} + 5x_{21} + 4x_{13} + 2x_{31} + 6x_{23} + x_{32}$$

$$\text{S.a } x_{12} + x_{21} = 1$$

$$x_{13} + x_{31} = 1$$

$$x_{23} + x_{32} = 1$$

$$x_{12} + x_{23} + x_{31} \leq 2$$

$$x_{13} + x_{32} + x_{21} \leq 2$$

$$0 \leq x_{ij} \leq 1 \text{ e inteiros}$$

$$\left. \begin{array}{l} i = 1, 2, 3 \\ j = 1, 2, 3 \end{array} \right\} i \neq j,$$

e a sua projeção:

$$\text{Min. } 2x_{12} + 2x_{13} + 5x_{23} + 8$$

$$\text{S.a } x_{12} + x_{23} - x_{13} \leq 1$$

$$-x_{12} - x_{23} + x_{13} \leq 0$$

$$x_{12} \leq 1$$

$$x_{23} \leq 1$$

$$x_{13} \leq 1$$

$$x_{12}, x_{23}, x_{13} \geq 0 \text{ e inteiros.}$$

De uma forma mais geral, o problema de Ordenação Linear pode ser formulado como:

$$\begin{array}{ll} \text{Min. } c^T x & \\ \text{(PI) S.a } Ax = b & \\ & x + s = e \\ & x, s \geq 0 \text{ e } x \text{ inteiro} \end{array}$$

onde a matriz A é formada pelas inequações triangulares 3-diciclo, mais as folgas. As componentes do vetor x são x_{ij} para $1 \leq i < j \leq p$, com respectivas folgas no vetor s e $e = (1, 1, \dots, 1)^T$ de dimensão apropriada.

A relaxação linear de (PI) é:

$$\begin{array}{ll} \text{(PL) Min. } c^T x & \\ \text{S.a } Ax = b & \\ & x + s = e \\ & x, s \geq 0 \end{array}$$

Para resolver o (PL) do problema discutido acima, Mitchell e Borches usaram o Método de Ponto Interior Primal-Dual de Redução de Potencial discutido na seção 2.3 deste trabalho.

Na próxima seção discutiremos como resolver o problema de Ordenação Linear usando o Método de Planos de Corte associado ao método de Pontos Interiores.

5.2 O Método de Planos de Cortes

A estratégia utilizada por [MB – 1996] para resolver o problema de ordenação linear é aplicar o método de Planos de Cortes com Pontos Interiores. Para tanto, é utilizada uma relaxação linear inicial do problema onde as restrições do tipo 3-diciclos são eliminadas. Em cada estágio do algoritmo de Planos de Cortes, uma relaxação do problema de Ordenação Linear é resolvida aproximadamente usando o método de Redução de Potencial Primal Dual. Se restrições do tipo 3-diciclo são violadas, um subconjunto destas restrições é adicionado à relaxação e o problema aumentado é resolvido. Procedemos desta maneira até que nenhuma restrição violada seja encontrada. Terminamos quando encontramos uma ordenação que está dentro da tolerância do limite inferior dada pelo valor da iteração dual corrente.

Eliminando as restrições do tipo 3-diciclos de (PL), a relaxação inicial do problema de Ordenação Linear assume a forma:

$$\begin{array}{ll} \text{Min. } c^T x & \\ (\text{PL}_0) \quad \text{S.a } x + s = e & \\ & x, s \geq 0. \end{array}$$

Este problema é chamado *relaxação primal inicial*.

A *relaxação dual inicial* é:

$$\begin{array}{ll} \text{Max } -e^T w & \\ (\text{PD}_0) \quad \text{S.a } z - w = c, & \\ & z, w \geq 0, \end{array}$$

onde cada um dos elementos de z e w é um par ordenado de vértices tal que $i < j$.

A solução ótima para (PL_0) e (PD_0) é dada por:

$$x_{ij} = \begin{cases} 1 & \text{se } c_{ij} \leq 0, \\ 0 & \text{caso contrário,} \end{cases} \quad s_{ij} = \begin{cases} 0 & \text{se } c_{ij} \leq 0, \\ 1 & \text{caso contrário,} \end{cases}$$

e

$$z_{ij} = \begin{cases} c_{ij} & \text{se } c_{ij} \geq 0, \\ 0 & \text{caso contrário,} \end{cases} \quad w_{ij} = \begin{cases} 0 & \text{se } c_{ij} \geq 0, \\ -c_{ij} & \text{caso contrário,} \end{cases}$$

Determinamos então restrições 3-diciclos que são violadas pela solução ótima da relaxação inicial, e adicionamos um subconjunto de restrições arcos disjuntos (dois diciclos são arcos disjuntos se eles não compartilham um arco). A relaxação passa então a ter a seguinte forma padrão:

$$\begin{array}{ll} \text{Min.} & c^T x \\ \text{S. a} & Ax = b \\ & x + s = u \\ & x, s \geq 0. \end{array} \quad (\text{PPL})$$

Aqui, A tem uma linha para cada restrição do tipo 3-diciclo adicionada (restrições (5.8) e (5.9)). Os elementos de b são 0 ou 1, apropriadamente. Cada linha tem 4 elementos diferentes de zero: três para os arcos do diciclo e um para a variável de folga. Seja m o número de linhas da matriz A . Assim, x tem um elemento diferente de zero para cada par ordenado de vértices $i < j$, e m elementos para as variáveis de folga, portanto o número de colunas de A é

$$n = m + \binom{p}{2}, \text{ onde } p \text{ é o número de vértices; a dimensão de } s \text{ é definida similarmente. Como}$$

ao vértice x foram incorporadas as variáveis de folga, temos que o limite superior de x será $u_i = 1$ se i corresponde a um arco e $u_i = 2$ se i corresponde a uma variável de folga, pois o maior valor que as variáveis de folga podem tomar na inequação (5.8) ou na inequação (5.9) é

2. A relaxação dual associada é:

$$\begin{array}{ll} \text{Max.} & b^T y - u^T w \\ \text{S. a} & A^T y + z - w = c \\ & z, w \geq 0 \\ & y \text{ irrestrito.} \end{array} \quad (\text{PDD})$$

A fim de se obter uma ordenação inicial, tomaremos o ordenação natural s , que será armazenado no vetor *order* e o valor da função objetivo associada em *upper*. Usaremos um

ponto no interior de $\bar{P}_2^p$ quando recentralizarmos o algoritmo depois de adicionar os planos de cortes. Este ponto é inicializado como $x^{feas} = 0.5e$, onde $e = (1, 1, \dots, 1)^T$.

Resolvendo a Relaxação antecipadamente

Uma das maneiras de melhorar a eficiência deste problema é usando a *terminação antecipada*. Neste caso, evitamos resolver a relaxação corrente até a otimalidade, reduzindo o número de iterações gastas na relaxação corrente. Além disso, a próxima iteração será mais centrada, o que significa que poucas iterações serão necessárias para a próxima relaxação.

Neste caso é possível procurar inequações válidas em cada iteração, porém se adicionarmos muitas inequações o problema torna-se de difícil resolução e pouco progresso é feito. Por isso, introduziremos um parâmetro através do *gap dual* de maneira que se o algoritmo está fornecendo muitas inequações, aumentamos sua tolerância τ_2 ; por outro lado, se poucas inequações são encontradas, diminuimos a tolerância. Como notamos, estaremos trabalhando com uma tolerância dinâmica. Porém, antes de procurar por inequações válidas 3-diciclo, o algoritmo executará outros testes, como veremos a seguir em sua descrição, nos passos 3 e 4.

Fixando uma ordenação

O vetor da melhor ordenação encontrado é armazenado em *order* e seu valor armazenado em *upper*. Podemos melhorar *order* arredondando a iteração corrente como segue. Se $x_{ij} > 0.6$, então estabelecemos que i é antes de j , se $x_{ij} < 0.4$, então estabelecemos j antes de i . Se este processo resulta em uma ordenação melhor que a melhor ordenação obtida até o momento, atualizamos *order* apropriadamente.

Também é necessário promover um “desempate” quando tentamos construir uma ordenação, devido à falta de preferências entre os pares. Isto é feito ocasionalmente.

Declara-se que uma solução é ótima quando encontramos uma ordenação que está dentro do gap dual relativo, de 10^{-8} . Há várias soluções ótimas para o problema de Ordenação Linear. A iteração primal x converge para uma combinação convexa destas soluções, mas as heurísticas usadas no algoritmo escolherão uma das ordenações.



Gerando Planos de Cortes

Quando geramos restrições do tipo 3-diciclo para adicionar ao problema de ordenação linear, o conjunto de restrições adicionadas gera apenas um elemento diferente de zero em cada coluna de A . Estas restrições contribuem para a estrutura da matriz A , ajudando na fatoração de Cholesky do produto AA^T , reduzindo o tempo gasto para calcular as projeções em cada iteração. Porém, temos a desvantagem de que com o aumento do número de restrições, aumenta o número de iterações, e esta desvantagem é mais significativa do que a vantagem de diminuir o tempo por iteração.

Com isso é aconselhável adicionar apenas um subconjunto destas restrições, classificados de acordo com a sua violação. Também é benéfico usar o um parâmetro τ_2 , aumentando o seu valor se a máxima violação está próxima de um e muitas restrições são violadas, evitando assim, que a relaxação corrente seja resolvida exatamente; e diminuindo seu valor se a máxima violação está próxima de zero e poucas restrições são violadas, pois neste caso a iteração corrente não está próxima o suficiente da solução ótima.

Atualizando um ponto em $\bar{P}_2^p$

O vetor x^{feas} é introduzido para armazenar pontos viáveis para $\bar{P}_2^p$ e deve satisfazer estritamente as restrições. O vetor x^{feas} é atualizado em cada iteração de x se x não é violado por nenhuma restrição do problema. Se x violar qualquer restrição, então tentamos atualizar x^{feas} tomando passos em direção à nova iteração x , como segue. Seja $d = x - x^{feas}$. Encontremos o maior valor de λ tal que $x^{feas} + \lambda d$ esteja em $\bar{P}_2^p$. Se este valor $\lambda_{\max}$ é no mínimo 0.1, atualizamos x^{feas} para $x^{feas} + 0.9\lambda_{\max}d$. Note que, $\lambda_{\max}$ pode ser encontrado eficientemente se o procurarmos simultaneamente quando encontramos planos de cortes violados por x .

Reiniciando o Algoritmo – Ponto de Partida

Usualmente, muitos planos de cortes são adicionados simultaneamente. Para simplificar a exposição, assumiremos que apenas uma restrição é adicionada.

Logo após a adição do plano de corte $a_0^T x \leq b_0$, a relaxação primal tem a forma:



$$\begin{aligned}
 & \text{Min } c^T x \\
 & \text{S. a } Ax = b, \\
 & \quad a_0^T x + x_0 = b_0, \\
 & \quad x + s = u, \\
 & \quad x_0 + s_0 = u_0, \\
 & \quad x, s \geq 0, \quad x_0, s_0 \geq 0,
 \end{aligned}
 \tag{PL2}$$

e a relaxação dual é:

$$\begin{aligned}
 & \text{Max } b^T y + b_0 y_0 - u^T w - u_0 w_0 \\
 & \text{S. a } A^T y + a_0 y_0 + z - w = c, \\
 & \quad y_0 + z_0 - w_0 = 0, \\
 & \quad z, w \geq 0 \quad z_0, w_0 \geq 0.
 \end{aligned}
 \tag{PD2}$$

O problema inicial tem uma restrição extra e a solução primal anterior é inviável para o problema corrente. A idéia é tentar reiniciar diretamente deste ponto. Isto é possível com o método de Redução de Potencial. Neste ponto, o algoritmo executa várias iterações até tornar-se viável, e às vezes pode ir para longe da solução ótima. Assim, é necessário resolver muitas iterações da relaxação corrente do *ponto de partida* como se tivéssemos partido de um ponto qualquer. Podemos melhorar consideravelmente o *ponto de partida* usando o ponto x^{feas} em $\bar{P}_2^p$. Normalmente, atualiza-se os primeiros $\begin{pmatrix} p \\ 2 \end{pmatrix}$ elementos de x para x^{feas} , e então as variáveis de folga x_0, s_0 , as variáveis de folga de x e os elementos de s são escolhidos para garantir que a nova iteração seja primal viável. Finalmente, todos os componentes muito pequenos de x são aumentados, e as correspondentes componentes de s são diminuídos na mesma proporção. Isto poderá causar uma perda (numericamente pequena) da viabilidade primal, mas melhorará a estabilidade numérica do algoritmo tão bem quanto nos dará uma iteração mais centrada.

A relaxação dual terá uma coluna extra e uma restrição correspondente a adição da variável de folga primal x_0 . A solução dual anterior é ainda viável no problema corrente, com $y_0 = 0$ e $w_0 = z_0$. A fim de ter um ponto interior, tomemos $w_0 = z_0 = 0.1$. E mais, todos os componentes pequenos de w são aumentados, com os correspondentes componentes de z aumentados na mesma quantia, a fim de manter a viabilidade dual. Esperamos que a nova restrição seja ativa na solução da nova relaxação, assim o valor ótimo de x_0 será zero. Se a

solução ótima é não degenerada, o valor ótimo de z_0 será positivo e o de w_0 será zero. Também é benéfico usar uma iteração de centralização imediatamente depois da adição dos planos de cortes.

Eliminando Restrições

Se um subconjunto das restrições não são eliminadas, a matriz de restrição torna-se muito grande e a fatoração de Cholesky de AA^T torna-se densa, assim o tempo gasto por iteração é proibitivo. Portanto, é necessário eliminar restrições que aparentam ser desnecessárias. Uma maneira de se fazer isso é se a variável de folga primal x_i correspondente a uma restrição maior que 0.4, então esta restrição não é importante, e podemos eliminá-la. Intuitivamente, a condição para decidir se eliminamos uma restrição dependerá das variáveis duais, assim como das variáveis primais.

Eliminar uma restrição não afeta a viabilidade primal, mas tornará a iteração dual inviável. Contudo, a perda da viabilidade dual é pequena e o algoritmo a recupera rapidamente.

O Algoritmo de Planos de Cortes

O algoritmo utilizado por [MB - 1996] está descrito a seguir. Denota-se por *maxadd* o parâmetro que controla o número máximo de planos de cortes adicionados ao problema, e por *lower*, o melhor valor da função objetivo do problema dual. Os demais parâmetros já foram abordados no texto acima.

Algoritmo de Planos de Cortes

Passo 1: (Inicialização). Resolva a relaxação inicial (PL_0). Do conjunto de restrições 3-diciclo violadas, escolha um subconjunto de 3-diciclos arco disjuntos e monte A , b , c e u , de acordo com as equações (5.7) a (5.9). Inicialize $x, s, y, w, z \in \Omega^0$ de acordo com a relaxação assim criada. Defina tolerâncias $\tau_1 = 10^{-6}$ (otimalidade), $\tau_2 = 0.3$ (geração de cortes) e $maxadd = 200$. Tome $lower = b^T y - u^T w$, *order* o vetor de incidência da ordenação natural, $upper = c^T x^{feas}$ e $x^{feas} = 0.5e$.



Passo 2: (*Resolvendo aproximadamente a relaxação*). Execute uma iteração do método de Pontos Interiores escolhido (neste caso, do método de Redução de Potencial). Seja $VFO^P = c^T x$, $VFO^D = b^T y - u^T w$ e $gap = VFO^P - VFO^D$.

a) Se a inviabilidade dual relativa é menor que 10^{-8} , seja $lower = \max\{lower, VFO^D\}$. Se $\frac{(upper - lower)}{\max\{1, |lower|\}} \leq \tau_1$, **pare**. A solução é dada por *order* e seu custo associado *upper*.

b) Se a inviabilidade dual relativa é maior que 10^{-8} ou se $\frac{gap}{\max\{1, |VFO^D|\}} > \tau_2$, repita este passo.

Passo 3: (*Atualizando order*). Use heurísticas para obter uma ordenação para x . Se esta ordenação é melhor que a melhor ordenação encontrado anteriormente, atualize *order* e *upper* apropriadamente.

Passo 4: (*Atualizando x^{feas}*). Seja $d = x - x^{feas}$. Seja $\lambda_{\max}$ o maior valor de λ tal que $x^{feas} + \lambda d$ satisfaça todas as inequações 3-diciclos. Se $\lambda_{\max} \geq 0.1$, então atualize $x^{feas} \leftarrow x^{feas} + 0.9\lambda_{\max} d$.

Passo 5: (*Procurando planos de corte*). Se x satisfaz todas as inequações 3-diciclos, diminua τ_2 e retorne ao passo 2.

Passo 6: (*Adicionando planos de corte*). Escolha inequações 3-diciclos pelo tamanho da violação. Tome as inequações violadas, adicione somente restrições de arcos disjuntos, adicione no máximo *maxadd*. Atualize τ_2 .

Passo 7: (*Eliminado planos de corte*). Examine as soluções primal e dual para certificar-se de que alguma restrição pode ser eliminada.

Passo 8: (*Atualizando x , s , y , w , z*). Faça $x = x^{feas}$. Atribua aos componentes adicionados de x correspondentes às variáveis de folga das restrições adicionadas os valores apropriados de modo a tornar x primal viável. Faça $s = u - x$. Atribua aos componentes adicionais de y o valor zero. Se alguma componente de x é menor que 10^{-6} , atribua-a 10^{-6} , diminua os componentes correspondentes de s de modo que $x + s = u$. Se alguma componente de z é



menor que 10^{-6} , atribua-a 10^{-6} , aumente os componentes correspondentes de w a fim de manter a viabilidade dual. Tome um passo de centralização. Atualize μ . Retorne ao passo 2.

Mitchell e Borches [MB - 1996] exibem uma comparação interessante. Eles resolveram uma série de problemas com uma implementação do método descrito feito em Fortran, numa SPARC 10/30 e compararam com os resultados obtidos por Grötschel *et alii* [GJR - 1984], que usou um IBM 370/168 e de Gerhard Reinelt, que usou uma SPARC 10/20, usando o código CPLEX. A comparação deles baseia-se em uma especulação acerca da velocidade relativa de cada máquina.

O comentário mais imediato a ser feito é sobre a razão de tempo entre o método de Pontos Interiores e o método Simplex. Apesar da superioridade do Simplex, observou-se claramente que ela cai com o aumento do tamanho do problema. Segundo Mitchell e Borches, espera-se que em problemas maiores o método de Pontos Interiores vença a disputa.

Uma melhoria considerável no algoritmo também poderia ser obtida com rotinas de álgebra linear especificamente desenhadas para o método de Pontos Interiores, já que em certos casos até 99% do tempo da execução é gasto nessas rotinas.

Os detalhes dos dados e dos resultados computacionais de cada problema (tamanho, cortes adicionados, iterações, etc.) podem ser encontrados nas referências citadas, mas para uma pequena ilustração, apresentamos na tabela 1 abaixo, retirada de [MB - 1996], o tempo entre a implementação de Mitchell e Borches (SPARC 10/30 com Pontos Interiores), e uma implementação de Gerhard Reinelt (SPARC 10/20 com CPLEX).

Vértices	SPARC 10/30	SPARC 10/20
44	21.4	3
56	75.8	20
60	176.8	50
79	3208.2	1944

Tabela 1: Comparação do tempo médio (em segundos).



Capítulo

6

Conclusões

Inicialmente devemos ressaltar que este trabalho teve um enfoque teórico a respeito do uso de Métodos de Pontos Interiores em Programação Inteira, embora discutíssemos alguns detalhes importantes a serem considerados na implementação computacional dos mesmos.

O método Primal-Dual abordado teve por base a estrutura sugerida por Wright [Wright – 1997]. Os métodos de Redução de Potencial e Trajetória Central citados nas seções 2.3 e 2.4, respectivamente, são variantes da estrutura definida na seção 2.2.2 – exceto pelo método de Pontos Interiores Inviável, o qual foi incluído devido à sua aplicação na reotimização dos problemas de (PI) – e se diferenciam entre si pela escolha do parâmetro de centralização e do tamanho do passo adotado em cada iteração do algoritmo. Estes métodos foram escolhidos a fim de ilustrar melhor a estrutura definida em 2.2.2 e pela suas aplicações em (PI). A extensão do método Primal-Dual para variáveis canalizadas foi incluída devido a necessidade de aplicá-lo em (PI), mais especificamente na resolução dos subproblemas que surgem nos métodos Branch-and-Bound e Branch-and-Cut.

O método de Planos de Cortes tem sido muito estudado e aplicado a problemas de Programação Inteira em geral. O interesse pelo Corte de Gomory surgiu do fato de ter sido o primeiro método a fornecer os fundamentos para uma teoria mais rigorosa [Dantzig - 1963] e, atualmente, junto com novas técnicas de Programação Linear e Programação Inteira tem despertado grande interesse [BCCN - 1996]. Outro fato importante é que o método de Planos de Cortes aplicados em cada nó da árvore do método Branch-and-Bound – o método Branch-and-Cut – tem melhorado muito a eficiência do algoritmo.

Os capítulos 3, 4 e 5 foram baseados principalmente nos trabalhos de Borches e Mitchell [BM - 1992] e [Mitchell - 1995]; Mitchell [Mitchell - 1997] e [Mitchell - 1995]; e Mitchell e Borches [MB - 1996], respectivamente. Toda a busca que foi feita nesta área de



pesquisa levaram-nos aos trabalhos deste grupo, embora não se tenha esgotado toda as fontes de busca. Gondzio [Gondzio – 1998], propõe o uso de métodos Primal-Dual que caminham através de um *Centro Analítico* para se gerar planos de cortes, porém não expõe como obter os planos de cortes, tratou apenas da questão de como obter um *ponto de partida* para os subproblemas após a adição de planos de cortes. Uma aplicação da técnica desenvolvida para resolver problemas de fluxo de multiprodutos não-lineares de grande porte (*large scale nonlinear multicommodity network flow problems*) é apresentada.

O uso de métodos de Pontos Interiores em Programação Inteira ainda é muito recente e tende a abrir um grande campo de pesquisa devido à frequência com que métodos de Pontos Interiores vêm sendo estudados. A sua vantagem esta associada ao fato que os métodos de Pontos Interiores começam a se mostrar úteis para problemas de grande porte, conforme as referências. O que se tem feito é usar métodos de Pontos Interiores em combinação com o método Simplex a fim de melhorar a performance do algoritmo de Planos de Cortes [Thienel - 1995]. Mas esta área ainda esta longe de se esgotar: há uma aposta tácita de que estes métodos podem ser competitivos com o Simplex, tudo depende de mais estudo.

[BM - 1992] B. Bonczak, J. E. Mitchell, "Using an Interior Point Method as a Branch and Bound Algorithm for Integer Programming" Technical Report 192, Mathematical Sciences Research Polytechnic Institute, Troy, NY 12180, March 1991, Revised July 2, 1992.

[Dantzig - 1963] G. B. Dantzig, *Linear Programming and Extensions*, Princeton, New Jersey, 1963.

[FP - 1963] S. C. Fang, S. Puthenpura, *Linear Optimization and Extensions: Theory and Algorithms*, Prentice Hall, New Jersey, 1963.

[FW - 1996] C. B. Ferreira, Y. Wajsbort, *Combinações Polédricas e Planos de Corte*, Faculdade de Engenharia, Instituto de Computação - Unicamp, Campinas, 1996.

[Friedlander - 1994] A. Friedlander, *Elementos de Programação Não-Linear*, Editora da Unicamp, 1994.

[GN - 1972] R. S. Garfinkel, G. L. Nemhauser, *Integer Programming*, Wiley, 1972.

[Gondzio - 1998] J. Gondzio, "Warm Start of the Primal-Dual Method Applied in the Cutting Plane Scheme" *Mathematical Programming* 83 (1998), 125- 143.



Bibliografia

- [AV - 1995] D. S. Atkinson, P. M. Vaidya. "A Cutting Plane Algorithm for Convex Programming that uses Analytic Centers". *Mathematical Programming* 69 (1995) 1 – 43.
- [Balas - 1979] E. Balas. "Branch – and – Bound – Implicit Enumeration". *Mathematical Programming* 71, pp. 185 – 191, 1979.
- [BCCN - 1996] E. Balas, S. Ceria, G. Cornuéjols e N. Natraj. "Gomory Cuts Revisited". *Operations Research Letters*, 19: 1 – 9, 1996.
- [BJ - 1977] M. S. Bazaraa, J. J. Jarvis. *Linear Programming and Network Flows*. John Wiley & Sons, USA, 1977.
- [BM - 1992] B. Borches, J. E. Mitchell. "Using an Interior Point Methods in a Branch and Bound Algorithm for Integer Programming". Technical Report 195, Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, March 1991. Revised July 7, 1992.
- [Dantzig - 1963] G. B. Dantzig. *Linear Programming and Extensions*. Princeton, New Jersey, 1963.
- [FP - 1993] S. C. Fang, S. Puthenpura. *Linear Optimization and Extensions: Theory and Algorithms*. Prentice Hall, New Jersey, 1993.
- [FW - 1996] C. E. Ferreira, Y. Wakabayashi. *Combinatória Poliédrica e Planos de Cortes Faciais*. 10^a Escola de Computação. Instituto de Computação - Unicamp, Campinas, 1996.
- [Friedlander - 1994] A. Friedlander. *Elementos de Programação Não-Linear*. Editora da Unicamp, 1994.
- [GN - 1972] R. S. Garfinkel, G. L. Nemhauser. *Integer Programming*. Wiley, 1972.
- [Gondzio -1998] J. Gondzio. "Warm Start of the Primal-Dual Method Applied in the Cutting Plane Scheme". *Mathematical Programming* 83 (1998), 125 – 143.



- [Gonzaga - 1989] C. C. Gonzaga. *Algoritmos de Pontos Interiores para Programação Linear*. 17^o Colóquio de Matemática, IMPA, CNPq, 1989.
- [Gomory - 1963] R. E. Gomory. "An Algorithm for Integer Solutions to Linear Programs". I. R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pp. 269-302. McGraw-Hill, New York, 1963.
- [Gomory - 1958] R. E. Gomory. "Outline of an Algorithm for Integer Linear Programs". *Bulletin of the American Mathematical Society*. 64, 275-278, 1958.
- [GJR - 1984] M. Grötschel, M. Jünger, G. Reinelt. "A Cutting Plane Algorithm for the Linear Ordering Problem". *Operations Research*, vol. 32, No. 6, November – December 1984, pp. 1195 – 1220.
- [HP - 1993] K. L. Hoffman, M. Padberg. "Solving Airline Crew Scheduling Problems by Branch-and-Bound". *Management Science*, vol. 39, No. 6, June 1993, pp. 657 – 682.
- [JN - 1992] E. L. Johnson, G. L. Nemhauser. "Recent Developments and Future Directions in Mathematical Programming". *IBM System Journal*, vol. 3, No. 31, 1992, pp. 79 – 93.
- [LD - 1960] A. Land, A. Doig. "An Automatic Method for Solving Discrete Programming Problems". *Econometrica*, 28, 497 – 520, 1960.
- [Levkovitz - 1992] R. Levkovitz. *An Investigation of Interior Point Methods for Large Scale Linear Programs: Theory and Computational Algorithms*. PhD Thesis, The University of West London, October, 1992.
- [LMSK - 1963] J. Little, K. G. Murty, D. W. Sweeney, C. Karel. "An Algorithm for the Traveling Salesman Problem". *Operation Research*, 11, 972 – 989, 1963.
- [Luenberger - 1989] D. G. Luenberger. *Linear and Nonlinear Programming*. 2^a edição, Addison – Wesley Publishing Company, USA, 1989.
- [MS - 1996] J. M. Martínez, S. A. Santos. *Métodos Computacionais de Otimização*. XIX Congresso Nacional de Matemática Aplicada e Computacional. 16 à 20 de setembro de 1996.
- [Matumoto - 1996] L. T. Matumoto. *Algoritmos de Pontos Interiores para Programação Linear e uma Extensão para a Programação Linear por Partes*. Dissertação de Mestrado, Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, 56 p., São Carlos, 1996.
- [Mitchell - 1997] J. E. Mitchell. "Fixing Variables and Generating Classical Cutting Planes when Using an Interior Point Branch and Cut Method to solve Integer Programming Problems". *European Journal of Operation Research*. 97: 139-148, 1997.



- [**Mitchell – 1997b**] J. E. Mitchell. “Computational Experience with an Interior Point Cutting Plane Algorithms”. Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, February 24, 1997.
- [**Mitchell - 1995**] J. E. Mitchell. “Interior Point Algorithms for Integer Programming”. Cap. 6 of the book: *Advances in Linear and Integer Programming*. Oxford University Press, 1995.
- [**MB - 1996**] J. E. Mitchell and B. Borchers. “Solving Real-World Linear Ordering Problems Using a Primal-Dual Interior Point Cutting Plane Method”. *Annals of Operations Research*, 62: 253 – 276, 1996.
- [**MPR - 1998**] J. E. Mitchell, M. P. Panos, M. G. C. Resende.. “Interior Point Methods for Combinatorial Optimization”. *Handbook of Combinatorial Optimization*, Kluwer Academic Publisher, 1998.
- [**NW - 1988**] G. L. Nemhauser, L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley, New York, 763 p., 1988.
- [**PR - 1991**] M. Padberg, G. Rinaldi. “A Branch-and-Cut Algorithm For The Resolution of Large-Scale Symmetric Traveling Salesman Problems”. *Siam Review*, vol. 33, No. 1, pp. 60–100, March 1991.
- [**Rangel - 1995**] M. S. N. Rangel. *Solving Integer Programming Problems Using Preprocessing and Cutting Planes: Theory and Implementation of Branch and Cut*. PhD Thesis, Brunel University, Uxbridge, U.K., 1995.
- [**Szwarcfiter - 1983**] J. L. Szwarcfiter. *Grafos e Algoritmos Computacionais*. Editora Campus, 2ª edição, 1983.
- [**Thienel - 1995**] S. Thienel. “ABACUS – A Branch-And-Cut System”. Tese de Doutorado, Universität zu Köln, Köln 1995.
- [**Wright - 1997**] S. J. Wright. *Primal-Dual Interior Point Methods*. SIAM, Philadelphia, 1997, 289 p.



