Métodos de *Branch and Bound*
*MS*777 - Professor Antônio Carlos Moretti

Flávia Barbosa 083552

1 Introdução

A programação inteira é um caso específico de problemas de programação linear em que a solução ótima só apresenta um significado prático se for um número inteiro. Tal problema pode ser colocado da seguinte forma:

$$\begin{aligned} \max \quad & z && (1)\\ \text{s.a} \quad & cx - z = 0 && (2)\\ & Ax \leq b && (3)\\ & x \geq 0 && (4)\\ & x_j \text{ inteiro} && (5) \end{aligned}$$

Onde:

- A é uma matriz $m \times n$,
- c é um vetor linha de dimensão n,
- b é um vetor linha de dimensão m,
- x é um vetor coluna de dimensão n de variáveis de decisão,

O presente trabalho irá discutir meios de obter a solução ótima para um problema de programação inteira a partir da solução do problema linear relaxado (sem a condição de integralidade).

2 Método *Branch and Bound*

O método de Branch and Bound é um método de busca em árvore que numera os pontos inteiros implicitamente, desde que haja uma quantidade finita deles, descartando os infactiveis e sempre acompanhando as soluções factíveis e os melhores valores para a função objetivo.

Considere o problema de maximização (1)-(5). O espaço das soluções

$$S = \{x : cx - z = 0, Ax \leq b, x \geq 0, x \text{ inteiro } \} \quad (6)$$

é particionado (ou ramificado) em sub-espaços

$$S^i = \left\{x^i : cx^i - z = 0, Ax^i \leq b, x^i \geq 0, x^i \text{ inteiro } \right\} \quad (7)$$

para eliminar as partes que espaço que não são factíveis para o problema inteiro. Assim, a coleção resultante de sub-problemas define cada ponto inteiro factível. O valor ótimo da função objetivo z^ipara cada sub-problema i criado determina um limitante superior para o valor da função objetivo

$$z = \max z_i \quad (8)$$

do problema original. Cada um dos sub-problemas criados são resolvidos: se a solução de um deles não é inteira será necessário ramificá-lo. A solução ótima é

alcançada quando o subproblema que apresenta que apresenta o maior limitante superior para o valor de z retorna uma solução inteira. Representamos em uma árvore esse procedimento de ramificação. Os nós da árvore representam cada um dos sub-problemas e os números associados a eles representam a ordem em que são analisados de acordo com o limitante superior.

Seja ,

$$x^* = x_1^*, ..., x_n^* \tag{9}$$

a solução ótima do problema linear sem as condições de integralidade e suponhamos que os valores de x_j^* não sejam inteiros. Sem perda de generalidade, podemos escolher x_1^* como o começo da árvore de busca. Seja, ainda,

$$\lfloor x_1^* \rfloor \tag{10}$$

e

$$\lceil x_1^* \rceil \tag{11}$$

o piso e o teto de x_1^* respectivamente. Então criamos dois novo problemas: um deles adicionando ao conjunto de restrições a desigualdade:

$$x_1^* \leq \lfloor x_1^* \rfloor \tag{12}$$

E um outro adicionando a desigualdade:

$$x_1^* \geq \lceil x_1^* \rceil \tag{13}$$

Resolvendo cada um desses problemas a árvore de busca será ramificada com a criaçãoo de duas novas folhas, uma contendo a soluçãao ótima z_1 e outra contendo a solução ótima z_2 associadas a cada um dos respectivos novos problemas.

Se a solução ótima obtida em algum desses problemas for inteira em todas as variáveis e o valor da função objetivo for melhor (no nosso caso, maior, já que temos um problema de maximização) então esse será o ótimo que procuravamos. Caso contrário, devemos fazer uma nova partição a partir da variável que não tiver retornado um valor inteiro, até que todas as variáveis assumam valores inteiros.

Se o problema original é limitado, temos a garantia de que a árvore desenvolvida será finita, desde que cada nó i de cada partição exija que uma dicotomia seja escolhida quando x_i não é inteiro.

2.1 Variáveis Binárias

O problema linear inteiro analisado anteriormente também aparece em alguns casos com uma variação: as variáveis de decisão devem assumir valores binários (0 ou 1). Para resolvê-lo, o mesmo raciocionio de ramificação é utilizado. O problema linear relaxado é resolvido e, a partir da solução obtida, novos subproblemas são criados adicionando as restrições $x_j = 1$ ou $x_j = 0$.

3 Método *Branch and Cut*

Esse método é uma generalização do método de *branch and bound*, uma vez que é contruído com a mesma lógica de *branch and bound* com a adição de vários cortes gerados e impostos em cada um dos nós da árvore de ramificação. Fazendo isso, esse método consegue um limitante mais exato para o nó antes de ramificá-lo e podá-lo. Algumas das técnicas usadas para gerar esses cortes são: arredondamento, disjunção e levatamento.

Na primeira delas (rounding), um limitante superior fracionário sobre uma variável inteira é arredondado para baixo e um limitante inferior fracionário é arredondando para cima. Essa técnica é usada na *redução do maior valor comum* (do inglês CGD - greatest common divisor), na qual a restrição é dividida pelo maior divisor comum inteiro dos coeficientes das variáveis e depois os novos coeficientes são arredondados usando essa lógica. Ou seja, a restrição

$$\sum a_j * x_j \leq b \tag{14}$$

é dividida por $c = \max a_j$ e, arredondando para baixo, obtemos o corte:

$$\sum \lfloor \frac{a_j}{c} \rfloor * x_j \leq \lfloor \frac{b}{c} \rfloor \tag{15}$$

Esse tipo de corte é muito fraco. Para gerar cortes mais fortes, precisa-se de informações adicionais sobre a estrutura do problema. Esses cortes são chamados de *cortes de arredondamento inteiro misto* (mixed integer rounding cuts, do inglês), que são cortes gerados por conjuntos envolvendo variáveis inteiras mistas. Como exemplo, podemos citar o *Corte de Gomory*

Já a técnica de disjunção (disjunction) é mais utilizada para construir cortes em problemas que involvem tanto variáveis continuas quanto variáveis inteiras.

Seja x_j uma variável fracionária. Partimos a região factivel em dois conjuntos separados adicionando umas das duas restrições abaixo:

$$x_1^* \leq \lfloor x_1^* \rfloor \tag{16}$$

ou

$$x_1^* \geq \lceil x_1^* \rceil \tag{17}$$

Cada uma dessas duas regiões é factivel para o problema original. Para combinar as duas regiões, aplicamos o método da disjunção para unir os dois conjuntos disjuntos. Com esse método, escolhemos dois pontos, um deles passando por $x_1^* = \lfloor x_1^* \rfloor$ e outro passando por $x_1^* = \lfloor x_1^* \rfloor$, unimos esses dois pontos por uma reta e determinando o sinal desigualdade de acordo com a factibilidade obtemos o corte disjunto. Esse corte é válido para o problema inteiro original porque todas o conjunto das soluções inteiras factiveis para o problema não é alterado.

Finalmente, o método de levantamento (lifting) é usado principalmente para problemas com variáveis binárias, e pode ser usado de duas maneiras: em uma delas, conhecida como *levantamento sequencial*, os coeficientes do corte são estimados um por um em uma sequencia independente de levantamento; na outra, esses coeficientessão estimados simultaneamente.

3.1 Corte Fracionário de Gomory

Suponha que o ótimo de um problema linear contenha um valor fracionário de alguma variável básica na linha r, ou seja:

$$x_{B_r} + \sum_{k \in J} \bar{a_{rk}} x_k = \bar{b_k} \tag{18}$$

onde J é o conjunto de indices das variáveis não básicas.

Essa equação pode ser usada para obter o corte fracionário de Gomory dado por:

$$\sum_{k \in J} f_{rk} x_k \geq f_{r0} \tag{19}$$

onde

$$f_{rk} = \bar{a_{rk}} - \lfloor \bar{a_{rk}} \rfloor \geq 0 \tag{20}$$

e

$$f_{r0} = \bar{b_r} - \lfloor \bar{b_r} \rfloor \geq 0 \tag{21}$$

3.2 Corte de Chvátal-Gomory

O corte de Gomory apresentado anteriormente é derivado da atualização dos coeficientes $\bar{a_j}$ do tableau ótimo do método simplex. Para encontrar o mesmo corte fracional mas em termos das variáveis e coeficientes originais, basta usar a relação:

$$\bar{a_j} = B^{-1} a_j \tag{22}$$

onde B^{-1} é o inverso da base ótima B, que pode ser extraída do tableau ótimo do simplex. Se o tableau inicial usa as variáveis de folga como as variáveis básicas, então os coeficientes associados a elas formam uma matriz identidade e, assim, temos a matriz atualizada $B^{-1}I = B^{-1}$. Esse resultado sugere que podemos encontrar B^{-1} das colunas alocadas abaixo das variáveis básicas no tableau correspondente sem calculos adicionais.

Considere os multiplicadores $u \geq 0$ e o poliedro

$$P = \left\{ x : \sum_j a_j x_j \leq b,\ x \geq 0 \right\} \tag{23}$$

definidos para o problema inteiro. Então a desigualdade:

$$\sum_j u^T a_j x_j \leq u^T b \tag{24}$$

é válida para P pois $u \geq 0$, $\sum_j a_j x_j \leq b$ e $x \geq 0$. Arredondando para baixo os coeficientes não inteiros do lado esquerdo da equação (24), obtemos a desigualdade válida:

$$\sum_j \lfloor u^T a_j \rfloor x_j \leq u^T b \tag{25}$$

O lado direito da equação (25) pode ser arredondado um pouco mais, uma vez que x_j é não negativo, resultando em:

$$\sum_j \lfloor u^T a_j \rfloor x_j \leq \lfloor u^T b \rfloor \tag{26}$$

A equação (26), definida $\forall u$, gera uma desigualdade válida para um problema inteiro puro.

4 Cortes para Problemas Inteiros Mistos

Na maioria das vezes, precisamos que apenas um grupo das variáveis de decisão retornem um resultado inteiro: temos, então, um problema de programação inteira mista, que pode ser de forma geral dado por:

$$\begin{aligned}
\max \quad & z && (27)\\
\text{s.a} \quad & cx + dy - z = 0 && (28)\\
& Ax \leq b_1 && (29)\\
& Gy \leq b_2 && (30)\\
& x, y \geq 0 && (31)\\
& x_j \text{ inteiro} && (32)
\end{aligned}$$

4.1 Corte de Gomory para Problema Inteiro Misto

Se temos o problema inteiro misto apresentado nas equações (27)-(32) e o ótimo do problema linear contém uma variável básica inteira com valor fracionário na linha r, então a linha r é selecionada para gerar um corte inteiro misto. Considere a linha gerada sendo:

$$\sum_j \bar{g_{rj}} y_j + \sum_k \bar{a_{rk}} x_k \leq \bar{b_r} \tag{33}$$

As variáveis continuas y_j são particionadas em dois casos: um com os coeficientes positivos e outro com os coeficientes negativos. As variáveis inteiras x_k também são particionadas em dois casos: um com

$$f_{rk} \leq f_{r0} \tag{34}$$

e outro com

$$f_{rk} > f_{r0} \tag{35}$$

onde f_{rk} e f_{r0} são dados pelas equações (20) e (21), respectivamente. Então, o corte pode ser gerado por:

$$\sum_{j:g_{rj}^{-}>0} g_{rj}^{-} y_j + \sum_{j:g_{rj}^{-}<0} g_{rj}^{-} y_j + \sum_{k:f_{rk}\leq f_{r0}} f_{rk} x_k + \sum_{k:f_{rk}>f_{r0}} \frac{f_{r0}(1-f_{rk})}{1-f_{r0}} x_k \geq f_{r0} \tag{36}$$

4.2 Corte de Arredondamento para Problema Inteiro Misto

Considere o problema linear inteiro misto dado pelas equações (27)-(32). A desigualdade:

$$x \leq \lfloor b \rfloor + \frac{y}{1-f} \tag{37}$$

é valida para o casco convexo de

$$S = \{(x, y) : x - y \leq b,\ x, y \geq 0,\ x \text{ inteiro}\} \tag{38}$$

onde $f = b - \lfloor b \rfloor$

4.3 Corte gerado pelo problema da mochila - *Knapsack Cover*

Considere a restrição do problema da mochila:

$$K = \left\{ x \in (0,1)^n : \sum_j a_j x_j \leq b,\ a_j > 0,\ b > 0 \right\} \tag{39}$$

Qualquer coeficiente não negativo pode ser convertido em um positivo substituindo x_j por:

$$x_j^{'} = 1 - x_j \tag{40}$$

Um conjunto C é uma *cobertura* se:

$$\sum_{j \in C} a_j > b \tag{41}$$

A cobertura é chamada de *minimal* se:

$$a_j \geq \sum_{j \in C} a_j - b > 0 \tag{42}$$

Então, se $C \subseteq N$ é uma cobertura de K e $|C|$ é o número de elementos em C, a desigualdade de cobertura:

$$\sum_{j \in C} x_j \leq |C| - 1 \tag{43}$$

é válida para K.

Além disso, a cobertura minimal definida pela equação (43) define uma faceta do casco convexo de K_c:

$$K_c = K \cap \{x : x_j = 0,\ j \in N/C\} \tag{44}$$

4.4 Corte gerado pelo problema da mochila - *Lifted Knapsack Cover*

Considere o conjunto K definido na equação (39). Seja M um subconjunto de N. Suponha que temos a desigualdade

$$\sum_{j \in M} \pi_j x_j \leq \pi_0 \tag{45}$$

válida para

$$K_c = K \cap \{x : x_j = 0,\ j \in N/M\} \tag{46}$$

Temos então um problema em que queremos encontrar os *coeficientes de levantamento* π_j tais que:

$$\sum_{j \in N} \pi_j x_j \leq \pi_0 \tag{47}$$

seja válida para K.

Usando o método do levantamento sequencial, os coeficientes π_j são obtidos um a um. Um dado coeficiente π_k é calculado de modo que a desigualdade:

$$\sum_{j \in M} \pi_j x_j + \pi_k x_k \leq \pi_0 \tag{48}$$

seja válida para $K_{M \bigcup \{k\}}$

4.5 Desigualdade de Cobertura *GUB* - Generalized Upper Bound

Esse tipo de desigualdade é obtido a partir do seguinte conjunto:

$$S = \left\{ y \in (0,1)^N : \sum_j a_j x_j \leq b,\ \sum_{j \in Q_i} x_j \leq 1,\ Q_i \cap Q_j = \phi\ \forall i \neq j,\ \cup_i Q_i = N \right\} \tag{49}$$

Um corte forte é dado por:

$$\sum_{j \in C} y_j \leq J \tag{50}$$

onde C é uma cobertura definida por dois elementos de C que pertencem ao mesmo Q_i.

5 Método *Branch and Price*

O método de *branch and price* é construido a partir do método de *branch and cut*, através da geração de colunas na árvore do *branch and cut* antes da ramificação, a qual ocorre quando mais nenhuma coluna proveitosa (que reduza o valor da função objetivo) pode ser encontrada e quando a solução do problema linear não satisfaz as condições de integralidade.

5.1 Geração de Colunas

Tal abordagem é usada quando a relaxação do problema linear contém muitas colunas (ou seja, muitas variáveis) para tratar explicita e simultaneamente. Assim, obtemos uma versão menor do problema original, que contém apenas um subconjunto de colunas (normalmente associadas às variáveis básicas) que são mantidas e atualizadas. Como a maioria das colunas irá ter a variável correspondente a ela igual a zero, somente as colunas vantajosas (associadas às variáveis não-básicas) são geradas e adicionadas ao problema original para melhorar a solução atual.

Semelhantemente à *Decomposição de Dantzig-Wolfe*, o problema é representado pelo método simplex do tableau revisado, que contém a base inversa e as soluções primal e dual. Então essa última é utilizada para atualizar a função objetivo de um subproblema, que é resolvido para determinar se a solução do problema linear é ótima e identificar uma coluna pivô para entrar na base melhorando o valor da solução caso ela ainda não seja ótima.

Se o ótimo encontrado satisfaz as condições de integralidade, então um limitante inferior é encontrado. Caso contrário, o valor ótimo fracionário pode ser utilizado como um limitante superior e, então, uma ramificação deve ser feita.

5.2 Decomposição de Dantizig-Wolfe

Essa decomposição utiliza o teorema de combinação convexa: um ponto y está em um poliedro

$$X = \{x : Ax = b, x \geq 0\} \tag{51}$$

convexo fechado e limitado se e somente se ele pode ser escrito como combinação convexa dos pontos extremos x_i desse poliedro, ou seja:

$$y = \sum_i \lambda_i * x_i \tag{52}$$

$$\lambda_i \geq 0 \tag{53}$$

$$\sum_i \lambda_i = 1 \tag{54}$$

Para a geração de colunas, vamos considerar que a solução inicial básica x_B, a base B e os coeficientes associados às variáveis básicas c_B são conhecidos. Considere também os multiplicadores do simplex associados a essa base: $\pi = c_B B^{-1}$. Para melhorar a solução factível básica, as colunas correspondentes às variáveis não básicas são superestimadas ('price out', do inglês) através da formação do custo dos coeficientes: $\bar{c_j} = c_j - \pi * a_j$. Se min $\bar{c_j} = c_s < 0$, então a solução atual pode ser melhorada se acrescentarmos s variável x_s à base com o uso do pivoteamento.

De modo mais geral, podemos escolher qual coluna deve entrar na base resolvendo o subproblema:

$$\min\ c(a_j) - \pi * a_j \tag{55}$$

onde $c(a_j)$ é o custo c em função de a_j.

Sendo $\pi = (\pi_1, \pi_0)$ os multiplicadores correspondentes às restrições

$$\sum_j a_j \lambda_j = b \tag{56}$$

e

$$\sum_j \lambda_j = 1 \tag{57}$$

respectivamente, podemos utilizar o algoritmo descrito a seguir para o desenvolvimento do método:

Passo 1: Usando os multiplicadores simplex π_1, resolver os subproblemas:

$$\begin{aligned} \min \quad & (c_i - \pi_1 a_i) x_i && (58)\\ \text{s.a} \quad & B_i x_i = b_i && (59)\\ & x \geq 0 && (60) \end{aligned}$$

e obter as soluções $x_i(\pi_1)$ e o valor ótimo da função objetivo z_i^0

Passo 2: Calcular:

$$\min\ \bar{f_j} = \sum_i z_i^0 - \pi_0 \tag{61}$$

Se min $\bar{f_j} \geq 0$, parar: a solução ótima é:

$$x^0 = \sum_{j \text{ básico}} \lambda_j x^j \tag{62}$$

Passo 3: Se $\min \bar{f}_j < 0$, gerar a coluna:

$$x = \begin{bmatrix} \sum_i a_i x_i(\pi_1) \\ 1 \end{bmatrix} \tag{63}$$

e, usando opivoteamento, obter a nova base inversa e o novo vetor de multiplicadores.

Voltar para o passo 1 e repetir.