CONTRIBUIÇÕES AO ESTUDO DO PROBLEMA DA MOCHILA

Vera Lucia da Motta

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

Prof. Nelson Maculan Filho
(Presidente)

Prof. Paulo O.Boaventura Netto

Prof. Jair Köiller

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 1981

DA MOTTA, VERA LUCIA

Contribuições ao Estudo do Problema da Mochila (Rio de Janeiro) 1981

IX, 199p. 29,7cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1981)

Tese - Univ. Fed. Rio de Janeiro. Fac. Engenharia

1. Estudos de Algoritmos de Programação Inteira a Uma Restrição I.COPPE/UFRJ II.Título (série)

AGRADECIMENTO

Agradeço especialmente ao professor Nelson Maculan Filho que, com seu encorojamento constante e orientação segura, tornou possível a realização deste trabalho. A dedicação e o carinho a mim dispensados, por voce Maculan, eu os guardo para devolver a tantos quantos de mim precisarem.

À professora e amiga Maria Luiza Villares agradeço o apoio que a mim dispensou, o qual contribuiu, sobremaneira, para que eu desenvolvesse com tranquilidade o meu trabalho.

Aos professores Jair Köiller e Paulo Oswaldo Boaventura agradeço a atenção dedicada ao meu trabalho, assim como orientação recebida.

Aos professores e colegas Antonio José de Nardi, Ronaldo Luiz Menezes de Albuquerque e Oswaldo de Souza Campos agradeço o incentivo que recebi desde o início do meu curso.

A excelente datilografia deste trabalho é devido à Sta. Suely Klajman a quem agradeço profundamente.

Finalmente o meu agradecimento ao Dr. Homero Roberto Passos Werneck de Carvalho, que me proporcionou a oportunidade de realizar meu curso, liberando-me das minhas atividades docentes da U.F.R.R.J e constantemente me incentivando com palavras de estímulo e confiança.

RESUMO

Neste trabalho nós enfocamos o problema da mochila, caracterizando sua importância, seus métodos de solução, técnicas e vantagens na transformação de um programa inteiro de várias restrições em um problema mochila, destacamos a necessidade de uma solução aproximada para socorrer problemas de grande porte e desenvolvemos um método de solução apoiado em grupos gerados por um determinado elemento e classes laterais.

É feito um estudo sobre o "cutting-stock problem", ressaltando a importância da técnica de geração de colunas, a qual resolve o embaraço causado pela quantidade exagerada de modelos de cortes. Essa técnica faz surgir a cada passo um problema mochila cuja rapidez de solução é fundamental para obtenção do ótimo do "cutting-stock problem". Desta feita a solução aproximada se torna a mais indicada. Usamos, por conseguinte, um algoritmo extremamente rápido quanto aproximado.

A técnica de programação dinâmica é desenvolvida e ilustrada com exemplos simples que nos permitem aplicar sem embaraços os algoritmos elaborados. Isto contribuiu para que se contruisse a cada passo um algoritmo mais eficiente computacionalmente. Neste caso a propriedade periódica é de relevante importância, pois minimiza, como concluímos, as computações.

A aplicação da teoria dos grupos no desenvolvimento de outra técnica de solução é precedida de uma caracterização algébrica dos elementos utilizados. Definições, proprie-

dades e teoremas da álgebra são amplamente abordados. Posteriormente à construção do algoritmo relativo a esta técnica, fazemos uma comparação entre os problemas mochila simples e o mochila grupo, dando ênfase à utilização de um ou outro processo de solução quanto ao aspecto operacional.

A importância do problema mochila ficou justificada pela sua utilização em várias situações práticas. O "cutting-stock problem" é a situação real escolhida para ressaltar a importância do problema, bem como a necessidade de busca de soluções extremamente rápidas para o problema mochila, o qual é usado como rotina muitíssimo solicitada no algoritmo elaborado.

O método dos multiplicadores de Lagrange dentro da abordagem considerável feita neste trabalho, se apresenta como método suficiente para redimensionar mochilas e estimar dados. A partir do momento em que sejam conhecidos os dados (daí não poder seu usado no "cutting-stock problem") mostramos que existem soluções fora da região viável que podem ser escolhidas com vantagens como solução aproximada. Caso exista uma solução fora da região viável mas suficientemente próxima de b pode-se pensar na possibilidade de alterar b (redimensionar a mochila).

ABSTRACT

In this work we approach the knapsack Problem ., and characterize its importance, solution methods and the advantages in the transformation of an integer program into a knapsack problem. We point out the need for approximate solutions to help solving large-scale problems and develop a solution method based on groups generated by an element and on lateral classes.

We study the "cutting-stock problem" and focus on the importance of column-generation, that solves the difficulties brought by the exagerate number of different cutting models. This technique gives rise at each step to a knapsack problem that must be solved by a fast method. This indicates the convenience of an approximate solution.

Dynamic programming technique is presented and illustrated with some simple examples fo tacilitate the comprehension of the proposed algorithms. We presented a series of increasingly computationaly efficient algorithms using besides other properties the periodice to minimize the number of the computations.

Theoretical group methods are then presented following analgebraic characterization of the involved elements. After an algorithm using this technique is built, we compare its performance in the solution of knapsack problem with that of a simple knapsack problem algorithm.

The knapsack problem has several practical. Among them we choose the cutting stock problem to show its relevance. For this last problem we develop a process which

demands extremely fast solutions of knapsack problems.Finally Lagrange multipliers method is proved to be a efficient method for the estimation of the data and to change if necessary the size of the knapsack. This is done when an almost feasible point which can be considered as an approximate solution is found.

ÍNDICE

	Páginas
CAPÍTULO I - INTRODUÇÃO	1
1. Formulação do Problema	2
2. Sua Importância	3
CAPÍTULO II - MÉTODOS DE SOLUÇÃO	7
1. Técnicas de Programação Dinâmica . . .	8
1.1. Problemas com variáveis limitadas.	9
1.2. Problemas com variáveis não limitadas	18
1.3. Algorítmos I e II	26
1.4. Uma Propriedade Periódica	40
1.5. Algorítmo III	46
2. Algoritmo de Branch and Bound	48
3. Método dos Multiplicadores de Lagrange	61
4. Comentários sobre os métodos apresentados	72
CAPÍTULO III - TRANSFORMAÇÃO DE UM PROBLEMA DE PROGRAMAÇÃO INTEIRA COM "m" RESTRIÇÕES EM UM PROBLEMA TIPO MOCHILA	74
1. O problema da transformação de um sistema de equações lineares em uma única equação linear	75
2. Teorema de Mathews; Processo de agregação I	77
3. Teorema de Glover; Processo de agregação II	89

Páginas

CAPÍTULO IV - TEORIA DOS GRUPOS EM PROGRAMAÇÃO INTEIRA A UMA RESTRIÇÃO 99

1. Introdução 100

1.1. Congruências em Z 100
1.2. Operação Binária; Grupo Abeliano . 105
1.3. Grupo Particular G(m) 107
1.4. Subgrupos ; Subgrupos de G(m) . . 107
1.5. Grupo Particular $G(m_1, m_2, .., m_n)$. 111
1.6. Isomorfismo 114
1.7. Classes Laterais 116

2. Formulação e Estudo do Problema Mochila Grupo 118

2.1. Formulação do Problema Mochila Grupo 118
2.2. Técnicas de solução 121

3. Formulação do Algoritmo para solução do Problema Mochila Grupo 144

4. Relação entre o Problema Mochila Simples e o Problema Mochila Grupo . . . 149

CAPÍTULO V - TRANSFORMAÇÃO DE UM PROBLEMA DE PROGRAMAÇÃO INTEIRA EM UM PROBLEMA GRUPO MOCHILA 159

1. Idéia do Processo 160

2. Suporte Teórico 161

3. Transformação I 162

4. Transformação II 166

5. Procedimento para obtenção da matriz $\hat{B}$ 168

6. Algoritmo para obtenção da matriz $\hat{B}$. 171

7. Caracterização do Processo 177

Páginas

CAPÍTULO VI - UMA APLICAÇÃO: "THE CUTTING STOCK PROBLEM" 180

1. Em que consiste o problema 181

2. Formulação do problema 182

3. Dificuldades apresentadas 183

4. Estudo para obtenção da solução a ser dada 185

5. Técnica aproximada para solução dos problemas mochila - Algoritmo Guloso. 192

REFERÊNCIAS BIBLIOGRÁFICAS 196

CAPÍTULO I

INTRODUÇÃO

1. Formulação do Problema

Consideremos a seguinte situação:

> "Uma transportadora suporta uma carga máxima "b". Existem n diferentes tipos de itens a serem transportados, sendo a_j e c_j respectivamente o peso e o valor relativo do item j. Queremos selecionar esses itens, de tal modo que a carga a ser transportada seja tal que a soma de todos os valores relativos é MÁXIMA".

A modelagem é simples. Seja x_j a quantidade do item j a ser transportada e sejam os demais parâmetros inteiros. Essas considerações concorrem para a formulação do seguinte problema de programação inteira:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} c_j \, x_j$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_j \, x_j \leq b$$

$$x_j \geq 0 \quad \text{e inteiro} \quad (j=1,2,\ldots,n)$$

onde a_j, c_j e b são inteiros.

O modelo acima, que resolve a situação criada, é do tipo:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} c_j \, x_j$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_j \, x_j \leq b$$

$$x_j \geq 0 \quad \text{e inteiro} \quad (j=1,2,\ldots,n) \qquad (1)$$

onde os c_j, a_j e b são inteiros, sendo que cada a_j $(j=1,2,\ldots,n)$ e b são positivos.

O problema de programação inteira com uma restrição (1) é chamado "Problema da Mochila". Este nome é devido ao fato dele estar relacionado com o enchimento de uma mochila que pode suportar um peso máximo b. O referido problema consiste em enchê-la de tal sorte que não seja excedida sua capacidade e se maximize a soma dos seus valores relativos c_j, sendo a_j o peso de cada item j. Supõe-se que existam n tipos de itens.

O problema da transportadora citado inicialmente é, portanto, um problema do tipo mochila.

Em muitas situações os c_j serão também positivos. Caso $c_j = 0$, o x_j correspondente é igualmente nulo, pois não há sentido transportar um item sem valor relativo, se queremos maximizar a soma desses valores relativos.

2. Sua Importância

Embora o problema tipo mochila seja o mais simples programa inteiro, ele merece estudo especial pois, além de ser representativo de muitas situações industriais, ele aparece em muitos algoritmos como subprograma. A busca de

soluções que tirem proveito da existência de apenas uma restrição fica portanto justificada.

Antes de passarmos ao capítulo seguinte, no qual estudaremos algoritmos específicos, nos permitiremos maior motivação fazendo uma modelagem tipo mochila ao analisarmos o clássico problema de investimento que segue:

"Existem n possibilidades de investimento independentes. O problema consiste da escolha entre essas n possibilidades de investimento de tal forma a maximizar o lucro total dos investimentos sujeito a restrição do capital disponível."

Assim sendo sejam:

c_j o lucro proveniente do projeto j

a_j o custo do projeto j

b o capital disponível

$x_j = 1 \rightarrow$ o projeto é aceito

$x_j = 0 \rightarrow$ o projeto é rejeitado

O modelo básico é então:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j = 0 \text{ ou } 1 \ , \ j=1,2,\ldots,n$$

o qual é um problema mochila com variáveis ZERO-UM. Os parâmetros c_j , a_j e b devem ser tomados como positivos e inteiros.

Para maior esclarecimento vale ressaltar aqui duas extensões para o modelo, assunto discutivo em Lorie and Savage |[21]| e Weingartner |[28]|.

1º caso: imaginemos a situação de investimento acima dividida em vários períodos com custos e capital disponível voltados para a exigência do período. Neste caso, temos:

a_{ij} (j=1,...,n ; t=1,...,T) → custo não negativo do projeto j no período t

b_t → capital disponível no período t (t=1,...,T)

Este caso é solucionado pelo modelo:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} c_j \, x_j$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_{tj} \, x_j \leq b_t \qquad (t=1,\ldots,T)$$

$$x_j = 0 \text{ ou } 1$$

O problema acima, caso $T \neq 1$, é um programa inteiro padrão ZERO-UM com uma matriz de restrição. Txn não negativa. Caso $T = 1$ torna-se um programa mochila.

2º caso: imaginemos que as n possibilidades de investimento são particionadas em subconjuntos disjuntos n_1,....., n_p ($p \leq n$) e é especificado que precisamente um projeto em cada subconjunto deve ser selecionado. Considerando ainda a divisão por período caracterizada no caso 1º, temos a seguinte solução:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} c_j \, x_j$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_{tj} \, x_j \leq b_t \qquad (t=1,\ldots,T)$$

$$\sum_{j \in n_i} x_j = 1 \qquad (i=1,\ldots,p \leq n)$$

$$x_j = 0 \text{ ou } 1$$

CAPÍTULO II

MÉTODOS DE SOLUÇÃO

1. Técnicas de Programação Dinâmica

Recordemos, inicialmente, a formulação do problema mochila para facilidade de referência:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} c_j \, x_j$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_j \, x_j \leq b$$

$$x_j \geq 0 \text{ e inteira}, \quad j=1,\ldots,n$$

onde os c_j, a_j e b são inteiros, sendo que cada a_j ($j=1,\ldots,n$) e b são positivos.

Definamos $f(k,g)$ como sendo o máximo valor da função objetivo usando os primeiros k e g, onde $k=1,\ldots,n$ e $g=0,1,\ldots,b$. Feito isto, temos que:

$$f(k,g) = \text{MÁXIMO} \quad \sum_{j=1}^{k} c_j \, x_j$$

$$\text{sujeito a:} \quad \sum_{j=1}^{k} a_j \, x_j \leq g$$

$$x_j \geq 0 \text{ e inteiro } (j=1,\ldots,k)$$

Podemos, portanto, encontrar $f(k,g)$ para $k=1,\ldots,n$ e $g=0,1,\ldots,b$.

Nosso objetivo é achar $f(n,b)$ onde n é o número de itens e b é o limite máximo de peso. Isto será conseguido encontrando-se todos os $f(k,g)$ a partir de $k=1,\ldots,n$ e $g=0,1,\ldots,b$. É fácil entender que $f(k,0)=0$, $\forall k$, pois se o limite de peso é ZERO então itens não podem ser considerados e evidentemente não há o que maximizar.

1.1. Problemas com variáveis limitadas

Embora $x_j \leq [g/a_j]$ para todo j, pois a positividade de $a_j (j=1,\ldots,n)$ e de g não permitem que x_j seja maior que aquele valor, em certos modelos torna-se necessária a restrição adicional $x_j \leq r_j \leq [g/a_j]$, r_j inteiro positivo. Como exemplo podemos citar o caso de um problema 0-1, onde $r_j = 1$.

Estudo da técnica de programação dinâmica aplicável a esses casos

Suponhamos que para um determinado k $(k=2,\ldots,n)$, os valores $f(k-1,g)$ são conhecidos para todo $g=0,1,\ldots,b$. Nosso intuito é achar $f(k,g)$ para um dado g. Para tal podemos começar escrevendo:

$$f(k,g) = \text{MÁXIMO} \left[c_k x_k + \sum_{j=1}^{k-1} c_j x_j \right]$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n-1} a_j x_j \leq g - a_k x_k$$

$$x_j \geq 0 \text{ e inteiro } (j=1,\ldots,k)$$

$$x_k = 0,1,\ldots, [g/a_k]$$

obs.: se um limite superior $r_k < [g/a_k]$ existe, então ele substituirá $[g/a_k]$ em $f(k,g)$ e assim $x_k \leq r_k$.

Para um valor inteiro conhecido x_k $(0 \leq x_k \leq [g/a_k])$, $f(k,g)$ reduz-se à forma abaixo, já que x_k deixa de ser incógnita.

$$f(k,g) = c_k x_k + \text{MÁXIMO} \left[\begin{array}{l} \sum_{j=1}^{k-1} c_j x_j \\ \\ \text{sujeito a:} \quad \sum_{j=1}^{n-1} a_j x_j \leq g - a_k x_k \\ \\ x_j \geq 0 \text{ e inteiro } (j=1,..,k-1) \end{array} \right]$$

Entretanto a expressão entre colchetes é precisamente $f(k-1, g-a_k x_k)$, a qual é conhecida visto que $g - a_k x_k$ é um inteiro não negativo não excedendo $\underline{b}$. Por exemplo: no cálculo de $f(2,4)$ aparece $f(1,4)$ que já foi calculado anteriormente. Podemos escrever, então:

$$f(k,g) = \text{MÁXIMO}_{x_k = 0,1,\ldots,[g/a_k]} (c_k x_k + f(k-1, g-a_k x_k)) \qquad (2)$$

Para cada $k = 2,3,\ldots,n$ a equação (2) pode ser usada para achar $f(k,g)$ para $g = 0,1,\ldots,b$.

obs.: 1) Quando $k = 0$ então $f(k,g) = 0$, pois não existem itens. Podemos então definir, sem perda de significado, $f(0,g) = 0$ e a equação (2) pode ser usada para $k = 1$, diretamente.

2) Quando $a_k > g \rightarrow [g/a_k] = 0$. Então, por (2) temos que:

$$f(k,g) = f(k-1,g) \quad \text{com} \quad x_k = 0$$

Lembrando que $f(k,0) = 0$, $\forall k$, podemos solucionar problemas com variáveis limitadas aplicando a equação (2), para cada $k=1,2,\ldots,n$ e tendo como condições iniciais:

$f(0,g) = 0$, $\forall g$

$f(k,0) = 0$, $\forall k$, com $x_k = 0$

EXEMPLO:

MAXIMIZAR $10x_1 + 16x_2 + x_3$

sujeito a: $2x_1 + 3x_2 + 2x_3 \leq 4$

$x_1 \leq 1$, $x_2 \leq 1$

$x_1, x_2, x_3 \geq 0$ e inteiro.

Como x_1 e x_2 não podem exceder 1, substituiremos $[g/a_k]$ por 1 sempre que $[g/a_k] \geq 1$, $k=1,2$. Para $k=3$ não há restrições, o limite é dado por $[g/a_3]$.

Condições iniciais: $f(0,g)=0$, $\forall g$

$f(k,0)=0$, $\forall k$, com $x_k = 0$

Todos os $f(k,g)$ serão obtidos, portanto, fazendo uso da equação (2).

(K = 1)

Temos $a_1 = 2$ e $c_1 = 10$

Obtenção dos limites para x_1 :

$$[g/a_1] = [g/2] = \begin{cases} 0 & \text{para } g = 0,1 \\ 1 & \text{para } g = 2,3,4 \end{cases}$$

Obtenção dos $f(1,g)$:

$f(1,0) = 0$ com $x_1 = 0$

$f(1,1) = 10.0 + f(0,1) \rightarrow f(1,1) = 0$ com $x_1 = 0$

$$f(1,g) = \text{MÁXIMO}(10x_1 + f(0,g - 2x_1)) = \text{MÁXIMO}(10x_1) = 10$$
$$x_1 = 0,1 \qquad\qquad x_1 = 0,1$$

$g = 2,3,4 \rightarrow f(1,2)=f(1,3)=f(1,4)=10$ com $x_1 = 1$

K = 2

Temos $a_2 = 3$ e $c_2 = 16$

Obtenção dos limites para x_2 :

$$[g/a_2] = [g/3] = \begin{cases} 0 & \text{para } g = 0,1,2 \\ 1 & \text{para } g = 3,4 \end{cases}$$

Obtenção dos $f(2,g)$:

$f(2,0) = 0$ com $x_2 = 0$

$f(2,g) = 16.0 + f(1,g-3.0) = f(1,g)$ para $g=1,2$ com $x_2 = 0$

$f(2,1) = 0$ com $x_2 = 0$

$f(2,2) = 10$

$f(2,g) = \text{MÁXIMO}(16x_2 + f(1,g-3x_2))$, $g = 3,4$
$x_2 = 0,1$

$f(2,3) = \text{MÁXIMO}(16x_2+f(1,3-3x_2))=\text{MÁXIMO}(10,16)=16$ com $x_2 = 1$
$x_2 = 0,1$

$f(2,4) = \text{MÁXIMO}(16x_2+f(1,4-3x_2))=\text{MÁXIMO}(10,16)=16$ com $x_2 = 1$
$x_2 = 0,1$

K = 3

Temos $a_3 = 2$ e $c_3 = 1$

Obtenção dos limites para x_3 :

$$[g/a_3] = [g/2] = \begin{cases} 0 & \text{para} \quad g = 0,1 \\ 1 & \text{para} \quad g = 2,3 \\ 2 & \text{para} \quad g = 4 \end{cases}$$

Obtenção dos $f(3,g)$:

$f(3,0) = 0$ com $x_3 = 0$

$f(3,1) = 1.0 + f(2,1-2.0) = f(2,1) = 0$ com $x_3 = 0$

$f(3,g) = \text{MÁXIMO}(1.x_3 + f(2,g-2x_3))$, $g = 2,3$
$x_2 = 0,1$

$= \text{MÁXIMO}(f(2,g), 1 + f(2,g-2))$, $g = 2,3$

Assim:

$f(3,2) = \text{MÁXIMO}(f(2,2), 1 + f(2,0)) = 10$ com $x_3 = 0$

$f(3,3) = \text{MÁXIMO}(f(2,3), 1 + f(2,1)) = 16$ com $x_3 = 0$

$f(3,4) = \text{MÁXIMO}(x_3 + f(2,4-2x_3)) = \text{MÁXIMO}(f(2,4), 1+f(2,2), 2+f(2,0))$
$x_3 = 0,1,2$

$= \text{MÁXIMO}(16, 11, 2) = 16$ com $x_3 = 0$

Formando um quadro com os valores encontrados temos:

g	f(1,g)	x_1	f(2,g)	x_2	f(3,g)	x_3
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	10	1	10	0	10	0
3	10	1	16	1	16	0
4	10	1	16	1	(16)	(0)

A solução ótima é $f(3,4) = 16$. Em princípio sa-

bemos apenas que $x_3 = 0$. Precisamos conhecer os valores das demais variáveis. É fácil percebermos que os valores das variáveis x_{k-1} são obtidos por intermédio de $f(k-1, g - a_k x_k)$. Esta conclusão é devida ao fato de considerarmos o problema sem o item K e evidentemente com um limite de peso diminuido do peso desses itens K , ou seja, $g - a_k x_k$. Esta idéia nos permite calcular as demais variáveis correspondentes a esse ÓTIMO obtido.

Como a solução ÓTIMA é $f(3,4) = 16$ com $x_3 = 0$, temos: $k = 3$ e $g = 4$. E assim:

$f(k-1,g-a_k x_k) = f(3-1,4-a_3.0) = f(2,4) = 16$ com $x_2 = 1$

Com o mesmo raciocínio, temos: $k = 2$ e $g = 4$ e então:

$f(k-1,g-a_k x_k) = f(2-1,4-a_2.1) = f(1,1) = 0$ com $x_1 = 0$

Logo a solução ÓTIMA para o problema é:

$f(3,4) = 16$ com $x_1 = x_3 = 0$ e $x_2 = 1$.

Caso em que a restrição é uma igualdade

Neste caso os valores $f(k,g)$ deverão ser computados de tal forma que o x_k correspondente seja tal que não permita folga na mochila. Assim sendo, o valor $f(k-1, g-a_k x_k)$ que aparece no cálculo de $f(k,g)$ só deverá ser computado se $g = a_k x_k$ para um dos possíveis valores de x_k. Caso contrário, x_k não será registrado e para efeito de cálculo, como veremos, será atribuido a $f(k,g)$ o valor $-\infty$. Desta forma $f(k,g) = -\infty$ significa não haver solução que evite a folga, ou seja, que satisfaça a igualdade, e por-

tanto o valor de x_k não é registrado.

O raciocínio acima nos torna possível resolver o problema fazendo uso da equação (2), desde que acrescentemos a condição inicial $f(0,g) = -\infty$, $g = 1,\ldots,b$ além de $f(k,0) = 0$, $\forall k$, com $x_k = 0$.

EXEMPLO:

$$\text{MAXIMIZAR} \quad 10x_1 + 16x_2 + x_3$$

$$\text{sujeito a:} \quad 2x_1 + 3x_2 + 2x_3 = 4$$

$$x_1 \leq 1 \ , \ x_2 \leq 1$$

$$x_1, x_2, x_3 \geq 0 \quad \text{e inteiro}$$

O cálculo dos limites para as variáveis x_1, x_2 e x_3 já foi efetuado no exemplo anterior e por conseguinte não será repetido.

Condições iniciais:

$$f(0,g) = -\infty \ , \ g=1,2,3,4$$

$$f(k,0) = 0 \ , \forall k, \text{com} \ x_k=0$$

(K=1)

Temos $a_1 = 2$ e $c_1 = 10$

$$x_1 = \begin{cases} 0 & \text{para } g = 0,1 \\ 1 & \text{para } g = 2,3,4 \end{cases}$$

Obtenção dos $f(1,g)$:

$f(1,0) = 0$ com $x_1 = 0$

$f(1,1) = 10.0 + f(0,1-2.0) = f(0,1) = -\infty$

$f(1,g) = \underset{x_1=0,1}{\text{MÁXIMO}}(10x_1 + f(0,g-2x_1)) \ , \quad g = 2,3,4$

Assim:

$f(1,2) = \text{MÁXIMO}(-\infty, 10) = 10 \quad \text{com} \quad x_1 = 1$

$f(1,3) = \text{MÁXIMO}(-\infty, 10+(-\infty)) = -\infty$

$f(1,4) = \text{MÁXIMO}(-\infty, 10+(-\infty)) = -\infty$

K = 2

Temos $a_2 = 3$ e $c_2 = 16$

$$x_2 = \begin{cases} 0 & \text{para } g = 0,1,2 \\ 1 & \text{para } g = 3,4 \end{cases}$$

Obtenção dos $f(2,g)$:

$f(2,0) = 0 \quad \text{com} \quad x_2 = 0$

$f(2,g) = f(1,g) \ , \ g = 1,2$

$f(2,1) = -\infty$

$f(2,2) = 10 \quad \text{com} \quad x_2 = 0$

$f(2,g) = \text{MÁXIMO}_{x_2=0,1}(16x_2 + f(1,g-3x_2))$

$= \text{MÁXIMO}(f(1,g), 16+f(1,g-3)) \ , \ g = 3,4$

$f(2,3) = \text{MÁXIMO}(-\infty, 16+0) = 16 \quad \text{com} \quad x_2 = 1$

$f(2,4) = \text{MÁXIMO}(-\infty, 16+(-\infty) = -\infty$

K = 3

Temos $a_3 = 2$ e $c_3 = 1$

$$x_3 = \begin{cases} 0 & \text{para } g = 0,1 \\ 1 & \text{para } g = 2,3 \\ 2 & \text{para } g = 4 \end{cases}$$

Obtenção dos $f(3,g)$:

$f(3,0) = 0 \quad \text{com} \quad x_3 = 0$

$f(3,1) = 1.0+f(2,1-2.0) = f(2.1) = -\infty$

$f(3,g) = \text{MÁXIMO}_{x_3=0,1}(x_3+f(2,g-2x_3))$

$= \text{MÁXIMO}(f(2,g),1+f(2,g-2),\ g = 2,3$

$f(3,2) = \text{MÁXIMO}(f(2,2),1+f(2,0)=10 \quad \text{com} \quad x_3 = 0$

$f(3,3) = \text{MÁXIMO}(f(2,3),1+f(2,1))=16 \quad \text{com} \quad x_3 = 0$

$f(3,4) = \text{MÁXIMO}_{x_3=0,1,2}(x_3+f(2,4-2x_3))$

$= \text{MÁXIMO}(f(2,4),1+f(2,2),2+f(2,0))=11 \quad \text{com} \quad x_3 = 1$

Formando um quadro com esses valores, temos:

g	f(1,g)	x_1	f(2,g)	x_2	f(3,g)	x_3
0	0	0	0	0	0	0
1	$-\infty$	.	$-\infty$	.	$-\infty$	.
2	10	1	10	0	10	0
3	$-\infty$	.	16	1	16	0
4	$-\infty$	.	$-\infty$	.	(11)	(1)

A solução ótima é $f(3,4) = 11$ com $x_3 = 1$. Obteremos a partir daí os valores ótimos para x_2 e x_1.

Obtenção de x_2 : (k_3 ; $g = 4$; $x_3 = 1$)

$f(k-1,\ g-a_k x_k)=f(2,4-2.1)=f(2,2)=10 \quad \text{com} \quad x_2 = 0$

Obtenção de x_1: ($k = 2$; $g = 2$; $x_2 = 0$)

$f(k-1,g-a_k x_k)=f(1,2-3.0)=f(1,2)=10 \quad \text{com} \quad x_1 = 1$

Solução ótima para o problema:

$$f(3,4) = 11$$
$$x_1 = x_3 = 1$$
$$x_2 = 0$$

OBSERVAÇÃO: Embora a técnica apresentada seja importante, visto que ela pode tratar problemas com variáveis limitadas, a mesma não tira vantagem da linearidade da função objetivo e da restrição. Ela pode envolver muitos cálculos, especialmente se b é grande e se as variáveis podem tomar muitos valores. Este procedimento é consideravelmente mais eficiente quando não se apresenta limites às variáveis, como veremos a seguir.

1.2. Problemas com variáveis não limitadas

Seja $f(k-1,g)$ conhecido para todo g e $k=2,...,n$. Nosso objetivo é calcular $f(k,g)$, a partir dessa informação. Como $f(k-1,g)$ é suposto conhecido podemos compará-lo com a expressão $c_k+f(k,g-a_k)$, que representa o valor de $f(k,g)$ para $x_k=1$ (por (2)). Pode ocorrer:

(i) $c_k+f(k,g-a_k) \leq f(k-1,g)$

(ii) $c_k+f(k,g-a_k) > f(k-1,g)$, $g \geq a_k$

Se ocorre (i) então não houve melhora nas buscas implicando em $x_k=0$ e $f(k,g)=f(k-1,g)$. Se ocorre (ii) significa que houve melhora e então x_k deve ser no mínimo igual a 1 (tal que não viole a restrição) e $f(k,g)= c_k + f(k,g-a_k)$.

Assim:

(i) $c_k+f(k,g-a_k) \leq f(k-1,g) \rightarrow x_k = 0$

(ii) $c_k+f(k,g-a_k) > f(k-1,g)$ para $g \geq a_k \rightarrow x_k \geq 1$

Como $f(k,g)$ é uma solução máxima, temos:

$$f(k,g) = \text{MÁXIMO}\{f(k-1,g), c_k+f(k,g-a_k)\} \text{ para } g \geq a_k \qquad (3)$$

A expressão (3) nos permite computar $f(k,g)$ para $k=1,\ldots,n$ e $g=0,1,\ldots,b$ com as condições iniciais citadas abaixo e já analisadas anteriormente.

$f(k,0) = 0$, $\forall k$, com $x_k = 0$

$f(0,g) = 0$, $\forall g$

EXEMPLO:

MAXIMIZAR $10x_1 + 16x_2 + x_3$

sujeito a: $2x_1 + 3x_2 + 2x_3 \leq 4$

$x_1, x_2, x_3 \geq 0$ e inteiros

K=1

Temos $a_1 = 2$ e $c_1 = 10$

$f(1,0) = 0$ com $x_1 = 0$

$f(1,g) = \text{MÁXIMO}\{f(0,g), 10+f(1,g-2) \text{ para } g \geq 2\}$, $g=1,2,3,4$

$f(1,1) = f(0,1)=0$ com $x_1 = 0$

$f(1,2) = \text{MÁXIMO}\{f(0,2), 10+f(1,0)\}=10$ com $x_1 \geq 1$

$f(1,3) = \text{MÁXIMO}\{f(0,3), 10+f(1,1)\}=10$ com $x_1 \geq 1$

$f(1,4) = \text{MÁXIMO}\{f(0,3), 10+f(1,2)\}=20$ com $x_1 \geq 1$

K = 2

Temos $a_2 = 3$ e $c_2 = 16$

$f(2,0) = 0$ com $x_2 = 0$

$f(2,g) = \text{MÁXIMO}\{f(1,g), 16+f(2,g-3) \text{ para } g \geq 3\}$

$f(2,1) = f(1,1) = 0$ com $x_2 = 0$

$f(2,2) = f(1,2) = 10$ com $x_2 = 0$

$f(2,3) = \text{MÁXIMO}\{f(1,3), 16+f(2,0)\} = 16$ com $x_2 \geq 1$

$f(2,4) = \text{MÁXIMO}\{f(1,4), 16+f(2,1)\} = 20$ com $x_2 = 0$

K = 3

Temos $a_3 = 2$ e $c_3 = 1$

$f(3,0) = 0$ com $x_3 = 0$

$f(3,g) = \text{MÁXIMO}\{f(2,g), 1+f(3,g-2) \text{ para } g \geq 2\}$

$f(3,1) = f(2,1) = 0$ com $x_3 = 0$

$f(3,2) = \text{MÁXIMO}\{f(2,2), 1+f(3,0)\} = 10$ com $x_3 = 0$

$f(3,3) = \text{MÁXIMO}\{f(2,3), 1+f(3,1)\} = 16$ com $x_3 = 0$

$f(3,4) = \text{MÁXIMO}\{f(2,4), 1+f(3,2)\} = 20$ com $x_3 = 0$

g	f(1,g)	x_1	f(2,g)	x_2	f(3,g)	x_3
0	0	(0)	0	0	0	0
1	0	0	0	0	0	0
2	10	(1)	10	0	10	0
3	10	1	16	1	16	0
4	20	(1)	20	(0)	20	(0)

As setas indicam as buscas horizontal e vertical efetuadas no processo de recuperação de solução ótima.

Recuperação dos Apontadores

Neste caso devemos examinar os valores de f(k,g), a partir de f(n,b). Se f(k,g) é tal que $x_k = 0$ isto significa que não houve melhora nas buscas e assim f(k,g) = f(k-1,g) e começamos o cálculo de x_{k-1} através f(k-1,g) , pois $x_k = 0$. Se no entanto $x_k = 1$, isto significa pelo que vimos ao determinar esta técnica, que x_k é no mínimo igual a 1. Assim sendo precisamos examinar o valor de x_k relativo a $f(k,g-a_k)$. Se for ZERO significa que x_k deve receber o valor 1, e passamos a examinar o valor de x_{k-1} através $f(k-1,g-a_k)$. Se ao contrário de $x_k = 0$ tivéssemos encontrado novamente $x_k = 1$, x_k seria incrementado de uma unidade e passaria então a valer 2. Nesse caso calculamos o novo $g = g-a_k$ (a proporção que encontramos $x_k = 1$, g é diminuido de a_k) e $f(k,g-a_k)$ é novamente examinado para verificar se x_k sofre outra incrementação (se for 1) ou se recebe o valor anterior (se for 0) nos permitindo analisar, então, x_{k-1} através $f(k-1,g-a_k)$, repetindo o mesmo raciocínio. Tendo por base a tabela estamos efetuando busca vertical quando subtraimos a_k de g e busca horizontal quando subtraimos 1 de k .

Este raciocínio nos conduz ao seguinte PROCEDIMENTO:

PASSO 1: Faça $k = n$, $g = b$, $j = 0$ e determine x_k a partir de f(k,g). Se $x_k = 0$ vá para o PASSO 2. Senão vá para o PASSO 3.

PASSO 2: Faça $k = k-1$. Se $k = 0$, PARE. Senão determine x_k a partir de f(k,g) e vá para o PASSO 3.

PASSO 3: Se $x_k = 1$ faça $g = g-a_k$, $j = j+1$ e vá para o PASSO 4. Se $x_k = 0$ faça $x_k = j$, $j = 0$ e vá para o PASSO 2.

PASSO 4: Determine x_k por $f(k,g)$ e volte ao PASSO 3.

APLICAÇÃO AO EXEMPLO

(1) $k = 3$; $g = 4$; $j = 0$

Como $x_3 = 0$ vá para o PASSO 2.

(2.1) $k = 3-1 \rightarrow k = 2$

$x_2 = 0$. Vá para o PASSO 3

(3.1) Como $x_2 = 0$ e $j = 0 \rightarrow x_2 = 0$ e vá para o PASSO 2

(2.2) $k = k - 1 \rightarrow k = 1$

$x_1 = 1$. Vá para o PASSO 3

(3.2) Como $x_1 = 1$ faça $g = g-a_1 \rightarrow g = 4-2 = 2$

$j = j+1 \rightarrow j = 1$ e vá para o PASSO 4

(4.1) $x_1 = 1$ e volte para o PASSO 3.

(3.3.1) Como $x_1 = 1$ faça $g = g - a_1 \rightarrow g = 0$

$j = j+1 \rightarrow j = 2$ e vá para o PASSO 4

(4.1.1) $x_1 = 0$ e volte para o PASSO 3.

(3.3.2) Como $x_1 = 0$ e $j = 2 \rightarrow x_1 = 2$, $j = 0$ e vá para o PASSO 2.

(2.3) $k = k-1 \rightarrow k = 0 \rightarrow$ PARE

Solução ótima: $f(3,4)=20$ com $x_3 = x_2 = 0$ e $x_1 = 2$

Caso em que a restrição é uma igualdade

Para resolver problemas desse tipo a única modificação natural se encontra nas condições iniciais. Como já analisamos anteriormente, devemos tornar $f(k,g) = -\infty$ e não registrar o valor correspondente de x_k sempre que a folga não puder ser evitada. Sendo assim o problema se resolve com a aplicação de (3) e das condições iniciais seguintes:

$f(k,0) = 0$, $\forall k$, com $x_k = 0$

$f(0,g) = -\infty$, $g = 1,2,\ldots,n$

EXEMPLO:

MAXIMIZAR $10x_1 + 16x_2 + x_3$

sujeito a: $2x_1 + 3x_2 + 2x_3 = 4$

$x_1, x_2, x_3 \geq 0$ e inteiros

Aproveitando os cálculos feitos no exemplo imediatamente anterior, temos:

(K = 1)

Temos $a_1 = 2$ e $c_1 = 10$

$f(1,0) = 0$ com $x_1 = 0$

$f(1,1) = -\infty$

$f(1,2) = 10$ com $x_1 \geq 1$

$f(1,3) = -\infty$

$f(1,4) = 20$ com $x_1 \geq 1$

K = 2

Temos $a_2 = 3$ e $c_2 = 16$

$f(2,0) = 0$ com $x_2 = 0$

$f(2,1) = -\infty$

$f(2,2) = 10$ com $x_2 = 0$

$f(2,3) = \text{MÁXIMO}\{f(1,3), 16+f(2,0)\} = 16$ com $x_2 \geq 1$

$f(2,4) = \text{MÁXIMO}\{f(1,4), 16+f(2,1)\} = 20$ com $x_2 = 0$

K = 3

Temos $a_3 = 2$ e $c_3 = 1$

$f(3,0) = 0$ com $x_3 = 0$

$f(3,1) = -\infty$

$f(3,2) = 10$ com $x_3 = 0$

$f(3,3) = \text{MÁXIMO}\{f(2,3), 1+f(3,1)\} = 16$ com $x_3 = 0$

$f(3,4) = \text{MÁXIMO}\{f(2,4), 1+f(3,2)\} = 20$ com $x_3 = 0$

g	f(1,g)	x_1	f(2,g)	x_2	f(3,g)	x_3
0	0	(0)	0	0	0	0
1	$-\infty$	.	$-\infty$	.	$-\infty$	.
2	10	(1)	10	0	10	0
3	$-\infty$	.	16	1	16	0
4	20	(1)	20	(0)	20	(0)

partir desse valor de k , g' é facilmente obtido e usando a fórmula (I) obtemos o valor de $F(b)$ e do x_1 correspondente, já que os demais x_j permanecem inalterados, segundo mostramos na secção anterior.

2. Um Algoritmo de Branch and Bound

Consideremos o programa linear:

$$\text{MAXIMIZAR} \quad \sum_{j=1}^{n} (c_j/a_j)y_j$$

$$\text{s.a.:} \quad \sum_{j=1}^{n} y_j = b$$

$$y_j \geq 0 \quad \text{e inteiro} \quad (J=1,\ldots,n)$$

onde $y_j = a_j x_j$, $j=1,\ldots,n$ e as variáveis estão ordenadas tais que $c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_n/a_n$. Isto acontecendo já sabemos que a solução do programa linear é $x_1 = b/a_1$, $x_j=0$ $(j\neq 1)$ e a função objetivo é $c_1(b/a_1)$.

Mostraremos que esse resultado se estende para o caso no qual as variáveis são limitadas.

1º caso: Existe limite superior inteiro

Chamemos esse limite superior inteiro de μ_j . Sendo assim $x_j \leq \mu_j$. Como $x_j = y_j/a_j$, temos:

$$y_j/a_j \leq \mu_j \rightarrow y_j \leq a_j\mu_j$$

(i) Quando $a_1\mu_1 > b$ concluimos que $y_1=b$ e $y_j=0$, $j\neq 1$ pois:

(1º) y_1 é a variável correspondente ao maior coefi