UNIDADE 7 – PROGRAMAÇÃO MATEMÁTICA

7.1) INTRODUÇÃO

Os problemas de Programação Linear Inteira podem ser entendidos como casos específicos da Programação Linear (conjunto solução contínuo), onde todas, ou parte, das variáveis de decisão devem ser inteiras.

Quando se usa esta classe de modelos é importante se ter mente o grau de dificuldade associado à sua solução. No entanto, isto não quer dizer que problemas que exijam computadores com alta capacidade computacional não possam ser resolvidos em um tempo aceitável. Mesmo que a solução ótima não seja encontrada, é possível obter boas soluções viáveis e mostrar quão próximo da solução ótima podem estar.

Um problema de programação linear inteira pode apresentar as seguintes situações:

- Todas as varáveis de decisões são inteiras: são problemas denominados Problemas de Programação Linear Inteira Pura – PLIP;
- Parte das varáveis de decisões são inteiras: são problemas denominados Problemas de Programação Linear Inteira Mista – PLIM;
- Todas as varáveis de decisões são binárias: são problemas denominados Problemas de Programação Linear Inteira Binária – PLIB;
- Parte das varáveis de decisões são binárias: são problemas denominados Problemas de Programação Linear Inteira Binária Mista – PLIBM.

O modelo formal pode ser expresso por:

$$\text{Max (ou Min)} \sum_{j=1}^{n} c_j x_j$$

$$sujeito\ a$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \ \ para\ \ i = 1,2,\dots,m$$

$$x_j \in I\ (ou\ Z)\ para\ \ j = 1,2,\dots,p\ (\leq n)$$

$$x_j \geq 0\ \ para\ \ j = p+1,\dots,n\ (\leq n)$$

7.2) FORMAS PARA RESOLVER PROBLEMAS PLI

Inicialmente, pode-se propor a solução de problemas de programação linear inteira por arredondamento ao final da aplicação de um método de programação linear, como por exemplo, pelo *Simplex*.

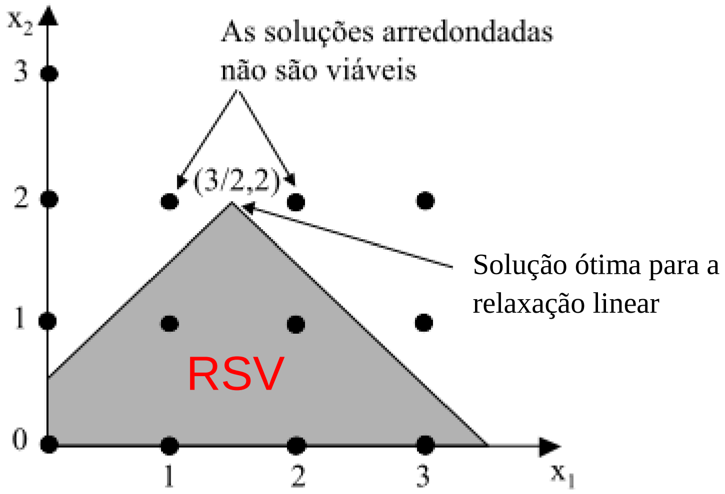
Para tanto, deve-se ignorar, temporariamente, a restrição que impõe que as variáveis de decisão devam ser inteiras. Caso a resposta não seja um número inteiro, deve-se arredondá-la. Uns dos maiores problemas desta forma de resolver problemas de PLI é que o arredondamento pode não redundar em uma solução ótima (ver figura adiante).

Outra abordagem é o método de enumeração que, pela avaliação das soluções viáveis, escolhe-se a melhor, ou seja, para problemas de maximização, a maior; para os de minimização, a menor. Um dos maiores entraves para a aplicação deste método é de ser impraticável para problemas reais que geralmente envolvem várias variáveis de decisão (observe o item 2 a seguir).

Devem-se considerar algumas observações:

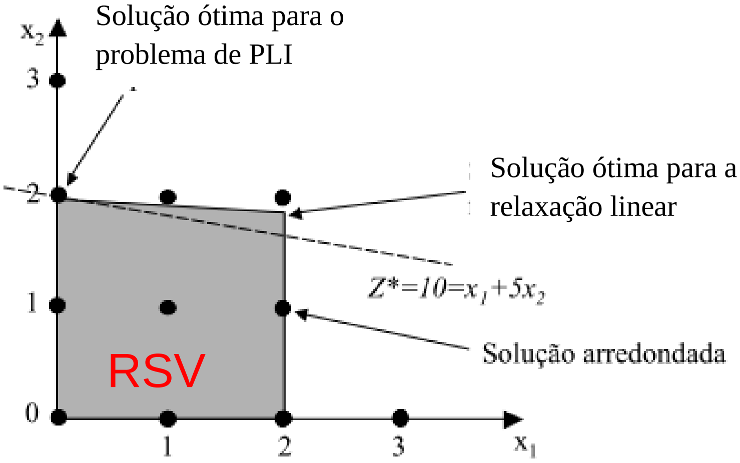
1) O número de soluções em um problema de PLI é finito, mas isto não implica que seja fácil de resolver;
2) Num problema de PLIB com *n* variáveis há 2^n soluções, por isso, para alguns problemas, fica impossível enumerar todas as soluções;
3) Os melhores algoritmos não podem garantir a solução de todos os problemas, mesmo relativamente pequenos (< 100 variáveis);
4) Para os valores que são suficientemente grandes para que o arredondamento não introduza erros significativos, pode-se até pensar neste artifício matemático;

   Para exemplificar, o gráfico a seguir expõe uma situação onde as soluções arredondadas não são viáveis. Em destaque a Região das Soluções Viáveis (RSV) de um problema de PL.



Fonte: Silva (2001)

O próximo gráfico apresenta uma situação em que a solução arredondada está longe da ótima:



Fonte: Silva (2001)

7.3) *BRANCH-AND-BOUND*

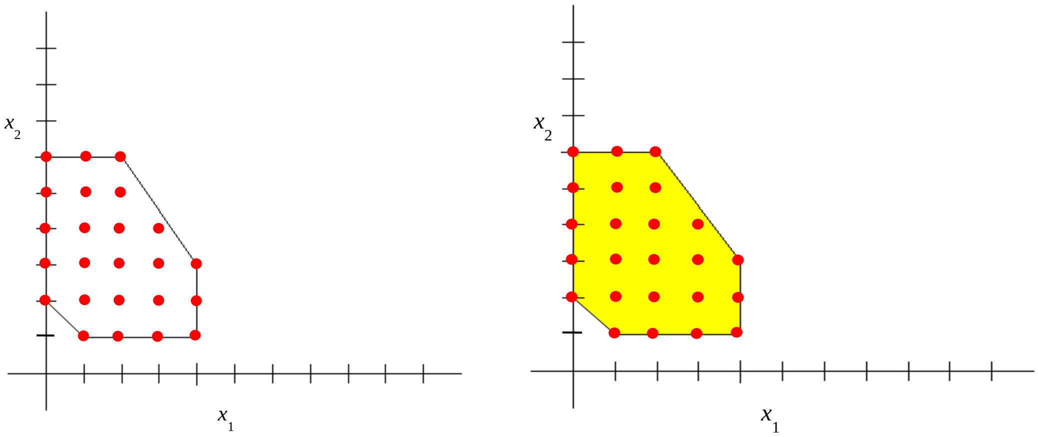
Como qualquer problema puro de PLI tem quantidade finita de soluções possíveis, deve-se considerar a utilização de um método de enumeração para encontrar um valor ótimo. Para esses casos, infelizmente a quantidade de possíveis soluções é, geralmente, muito grande, sendo então fundamental que o método utilizado seja suficientemente estruturado para que apenas uma pequena parte das soluções possíveis sejam realmente examinadas.

O método *Branch-and-Bound* (em português, particionar e limitar “as partições”) é um algoritmo que apresenta essa qualidade. Como os problemas de PLI são “relativamente grandes”, para resolvê-los diretamente deve-se dividi-lo em sub-problemas cada vez menores, até que estes possam ser solucionados. Sendo assim, a ideia é desenvolver uma enumeração inteligente dos pontos candidatos (nós) em busca da solução ótima inteira do problema, por meio da partição do espaço e avaliação progressiva das soluções.

A forma de divisão em problemas menores parte do princípio da separação de uma das variáveis de decisão inteiras, em um problema relaxado, utilizando-a em restrições contraditórias, criando uma espécie de ramificação (a partir de um nó), como em uma árvore.

Uma das formas de relaxação consiste em, temporariamente, ignorar as restrições de integralidade do problema de PLI, tornando-o um problema de PL, ficando, portanto, mais simples de resolver. A partir deste, pode-se usar para resolvê-lo o método *Simplex*. Deve-se considerar que o conjunto de soluções viáveis do problema original (PLI) esteja contido no conjunto de soluções viáveis do problema relaxado (PL), como exemplificada na figura adiante, implicando em:

a) Se o problema relaxado não tem solução viável, então o problema de PLI também não tem;
b) O valor mínimo do problema de PLI não é menor que o valor máximo do problema relaxado;
c) Se uma solução ótima do problema relaxado é viável no problema de PLI, então ela é uma solução ótima do problema de PLI.
d)

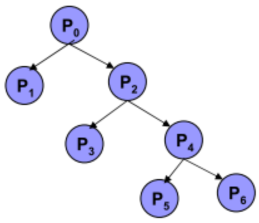
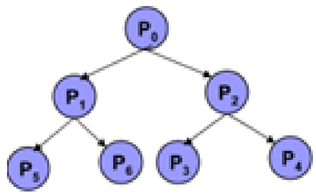


À esquerda, o conjunto de soluções de um problema de PLI e à direita, a RSV de um problema de PL. Fonte: Silva (s/d)

A escolha do ponto (nó) para ramificação da árvore pode-se ser efetuada, dentre as várias técnicas, nas seguintes:

- *Jumptracking*: implementa uma busca em largura (figura a seguir), onde um nó com o mínimo limite inferior é selecionado para examinação. Nesta estratégia o processo de ramificação salta de um ramo para outro na arvore de busca.
- *Backtracking*: implementa a busca em profundidade (figura a seguir), onde os nós descendentes de um nó pai são examinados em uma ordem arbitrária ou em ordem de limites inferiores não-decrescentes. Nesta estratégia, primeiramente prossegue-se até o nível mais baixo por algum caminho para encontrar uma solução tentativa e então refazer aquele caminho para cima até o primeiro nível com nós ativos e assim por diante.

É fácil notar que a estratégia *jumptracking* tende a construir uma grande lista de nós ativos, enquanto *backtracking* mantém relativamente uns poucos nós na lista a qualquer momento. Uma vantagem do *jumptracking* é a qualidade de suas soluções tentativas, que são geralmente muito mais próximas do ótimo do que soluções geradas por *backtracking*, especialmente nos estágios iniciais da busca.

<table>
<tr><td>Busca em profundidade
(Backtracking)

</td><td>Busca em Largura
(Jumptracking)

</td></tr>
<tr><td colspan="2">Variantes Híbridas</td></tr>
</table>

Na análise dos pontos candidatos faz-se necessário determinar quais são os pontos promissores, ou seja, aqueles que devam ser examinados ou descartados para análises futuras. Esta análise segue o seguinte critério:

e) O <u>problema candidato relaxado</u> (PL) não tem solução viável. Devido ao item *a* anterior, o <u>problema candidato</u> (PLI) também não tem solução viável;
f) A solução ótima do <u>problema candidato relaxado</u> é pior do que a melhor solução atualmente conhecida. Observar que a solução ótima do

problema candidato relaxado é sempre melhor ou igual à solução do problema candidato e de seus descendentes.
e.1) Num problema de maximização, o máximo do problema relaxado constitui o limite superior para o máximo do problema original;
e.2) Num problema de minimização, o mínimo do problema relaxado constitui limite inferior para o mínimo do problema original.

g) Uma solução ótima do problema relaxado é viável, também é no problema candidato. Devido ao item *c* anterior, ela também é ótima no problema candidato. Como uma solução viável de qualquer dos sub-problemas é também uma solução viável do problema, então a solução é também factível. Caso a solução seja melhor que a atual, a solução deste problema ocupará a posição de melhor solução atual, descartando a anterior.

O próximo exemplo, exposto por Silva (2001), apresenta a aplicação do método *Branch-and-Bound* para variáveis de decisão binárias (PLIB).

Exemplo 1:

$Max\ Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$
$sujeito\ a$
$6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10$
$x_3 + x_4 \leq 1$
$-x_1 + x_3 \leq 0$
$-x_2 + x_4 \leq 0$
x_j é $binário\ para\ j = 1,2,3,4$

Solução:

- ✓ Quando se lida com variáveis binárias, a forma mais simples de particionar o problema é fixar o valor de uma das variáveis, como por exemplo, x_1=0 e x_1=1
- ✓ Fazendo-se a substituição de x_1 no problema inicial obtém-se dois novos sub-problemas. Estes novos problemas são mais simples (ou menores...) do que o inicial.

Para x_1=0

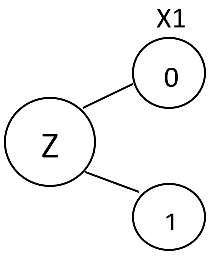
$Max\ Z = 5x_2 + 6x_3 + 4x_4$
$sujeito\ a$
$3x_2 + 5x_3 + 2x_4 \leq 10$
$x_3 + x_4 \leq 1$
$x_3 \leq 0$
$-x_2 + x_4 \leq 0$

Para x_1=1

$Max\ Z = 9 + 5x_2 + 6x_3 + 4x_4$
$sujeito\ a$
$3x_2 + 5x_3 + 2x_4 \leq 4$
$x_3 + x_4 \leq 1$
$x_3 \leq 0$
$-x_2 + x_4 \leq 0$

$$x_j \text{ é binário para } j = 2,3,4 \qquad x_j \text{ é binário para } j = 2,3,4$$

Para os sub-problemas anteriores, pode-se estruturar uma árvore denominada de solução ou de enumeração. A figura a seguir apresenta esta árvore.

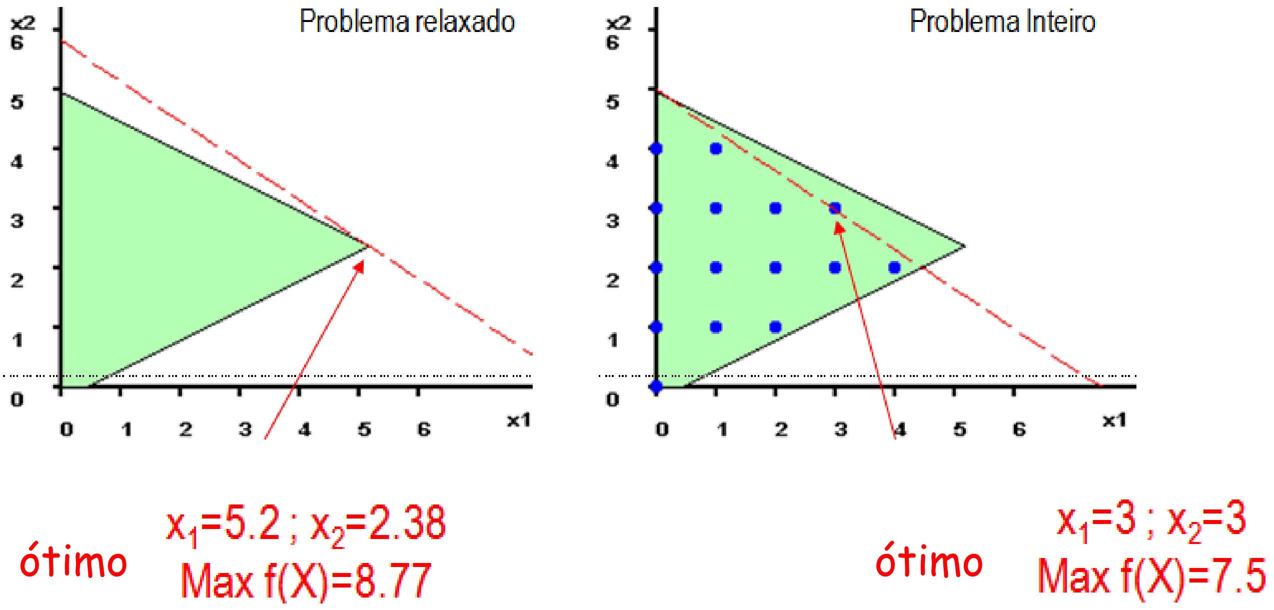


Existem muitos métodos sofisticados para se fazer esta ramificação, embora neste exemplo fosse considerada a escolha das variáveis na sua ordem natural.

Para cada um desses sub-problemas faz-se necessário calcular um limite à qualidade da sua melhor solução. Isto é geralmente resolvido por uma versão simplificada (relaxada) do problema. Esta relaxação é geralmente obtida eliminando uma ou mais restrições de integralidade do problema.

Por exemplo:

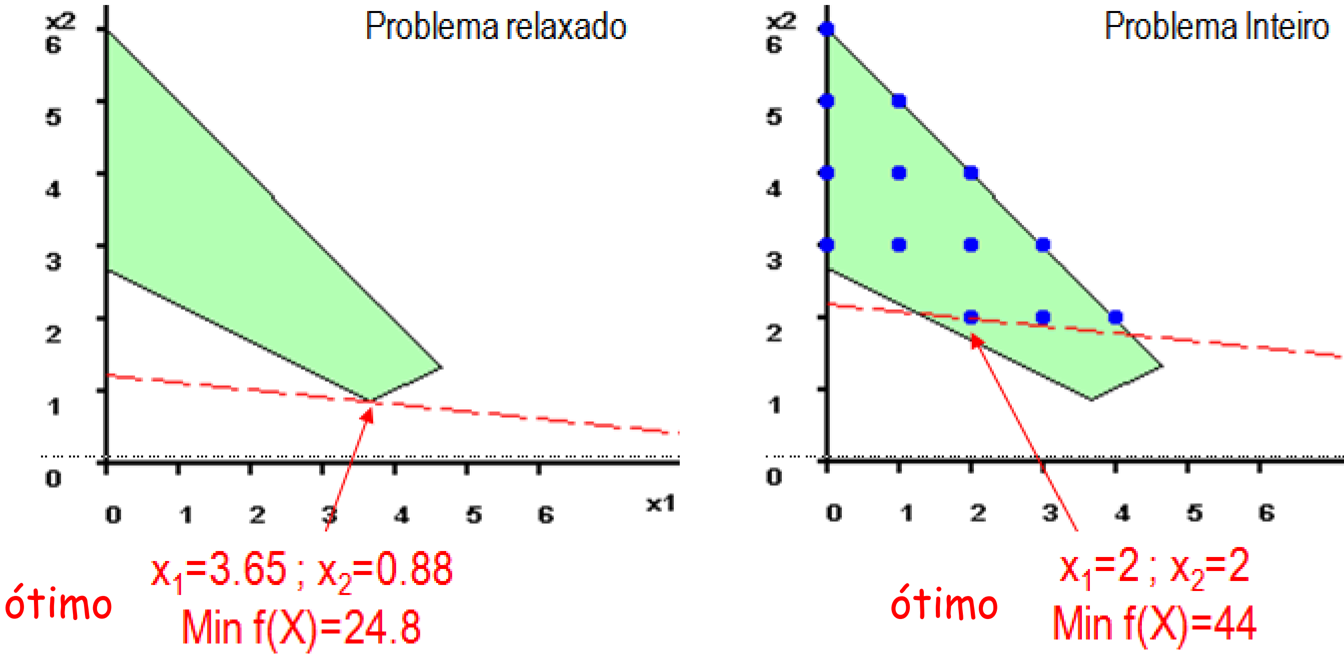
a) Para o caso de maximização, conforme o item e.1 anterior:



$f(X) \leq 8.77$

Fonte: Silva (s/d)

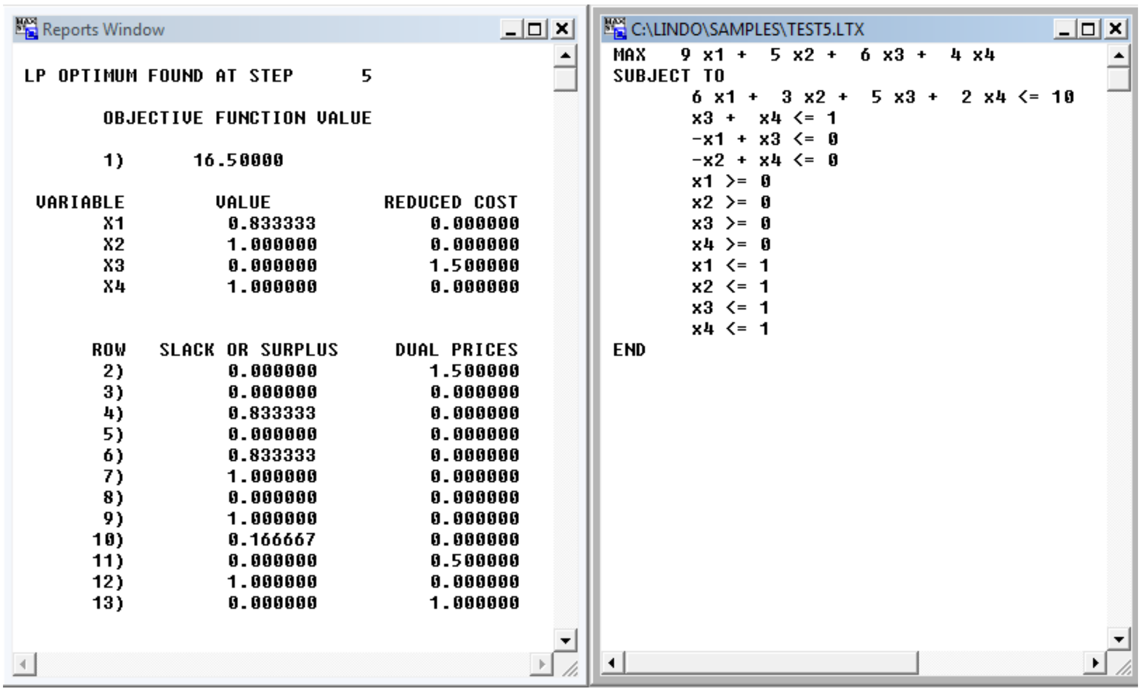
b) Para o caso de minimização, conforme o item e.2 anterior:



$f(X) \geq 24.8$

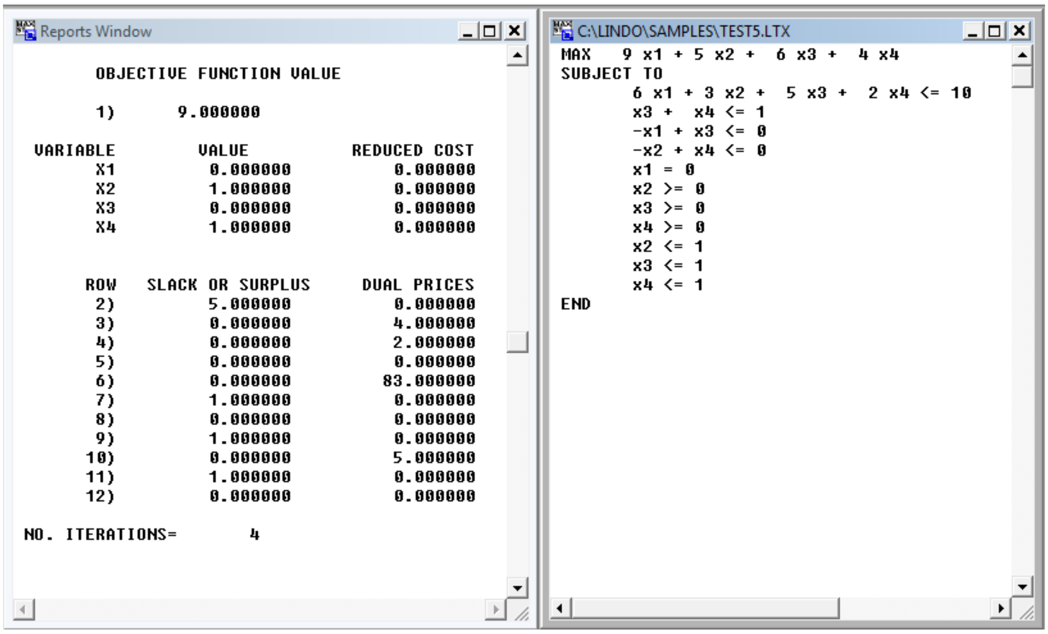
Fonte: Silva (s/d)

Voltando-se para o problema, para se obter a <u>solução inicial relaxada</u> do problema de PLIB, substitui-se a restrição natural (última linha do modelo) por $0 \leq x_j \leq 1$. Resolvendo-se pelo método *Simplex*, usando o software Lindo (figura a seguir) chega-se ao seguinte resultado: x_1=0,83; x_2=1;x_3=0;x_4=1; Z=16,5

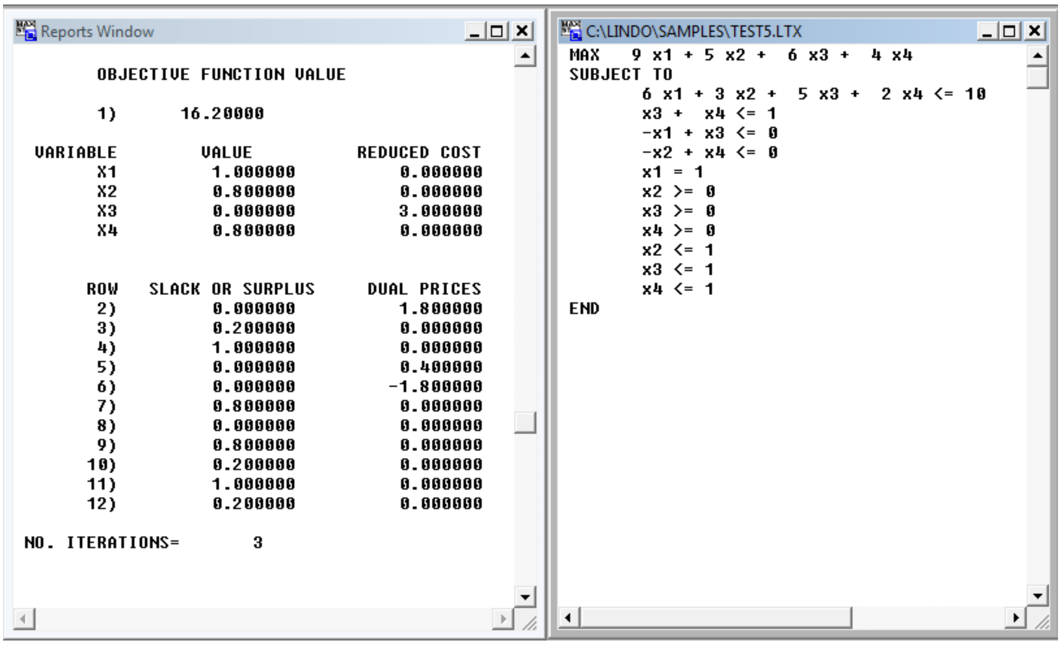


Portanto Z deve ser menor ou igual a 16,5. Como Z deve ser um número inteiro, arredonda-se a solução para 16. Tomando-se agora os dois sub-problemas (x_1=0 e x_1=1) propostos anteriormente, chega-se a seguinte solução:

Para x_1=0: x_2=1; x_3=0; x_4=1; Z=9 (Figura a seguir com a solução no software Lindo). Portanto Z ≤ 9



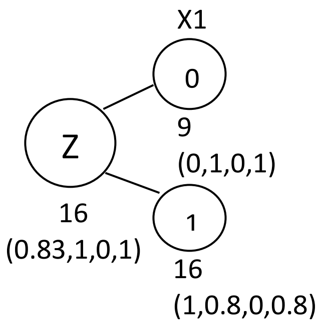
Para x_1=1: x_2=0,8; x_3=0; x_4=0,8; Z=16,2 (Figura a seguir com a solução no software Lindo). Portanto Z ≤ 16 (valor inteiro).



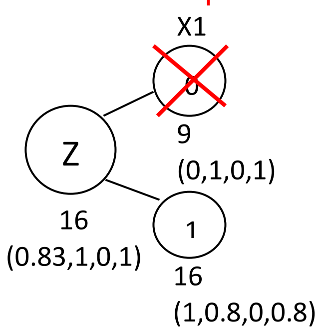
```
Reports Window

        OBJECTIVE FUNCTION VALUE

        1)      16.20000

  VARIABLE        VALUE          REDUCED COST
        X1         1.000000          0.000000
        X2         0.800000          0.000000
        X3         0.000000          3.000000
        X4         0.800000          0.000000


       ROW   SLACK OR SURPLUS     DUAL PRICES
        2)         0.000000          1.800000
        3)         0.200000          0.000000
        4)         1.000000          0.000000
        5)         0.000000          0.400000
        6)         0.000000         -1.800000
        7)         0.800000          0.000000
        8)         0.000000          0.000000
        9)         0.800000          0.000000
       10)         0.200000          0.000000
       11)         1.000000          0.000000
       12)         0.200000          0.000000

 NO. ITERATIONS=       3
```

```
C:\LINDO\SAMPLES\TEST5.LTX
MAX    9 x1 + 5 x2 +  6 x3 +  4 x4
SUBJECT TO
         6 x1 + 3 x2 +  5 x3 +  2 x4 <= 10
         x3 +  x4 <= 1
         -x1 + x3 <= 0
         -x2 + x4 <= 0
         x1 = 1
         x2 >= 0
         x3 >= 0
         x4 >= 0
         x2 <= 1
         x3 <= 1
         x4 <= 1
END
```

A solução em árvores expressa os resultados dos sub-problemas anteriores.

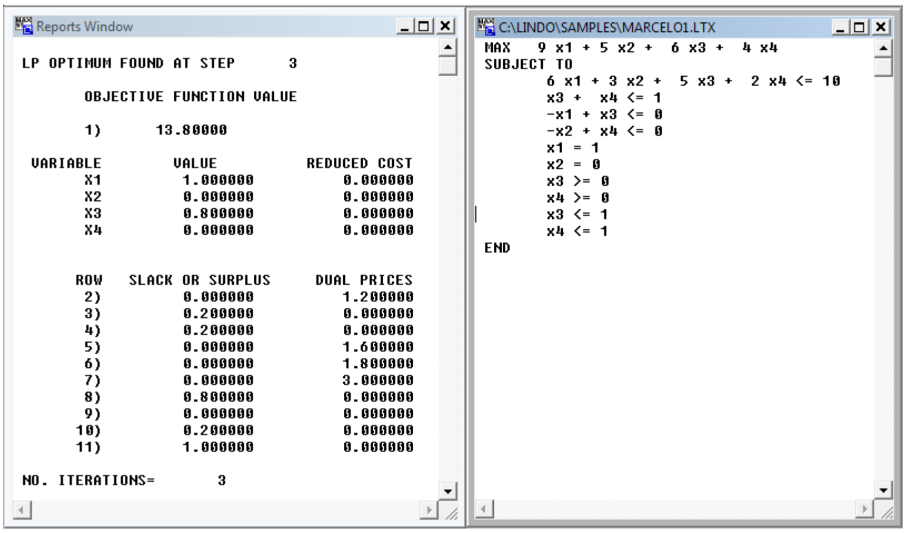


A solução relaxada do nó x_1=0 é inteira, portanto, esta deve fazer parte como solução ótima do sub-problema em questão e candidata para solução ótima final (Z*=9). Como 9 é o valor máximo neste ramo, as outras soluções derivadas conduzirão à respostas inferiores, **<u>o que não é interessante</u>**. Sendo assim, este ramo não deverá servir para a continuidade da solução do problema.

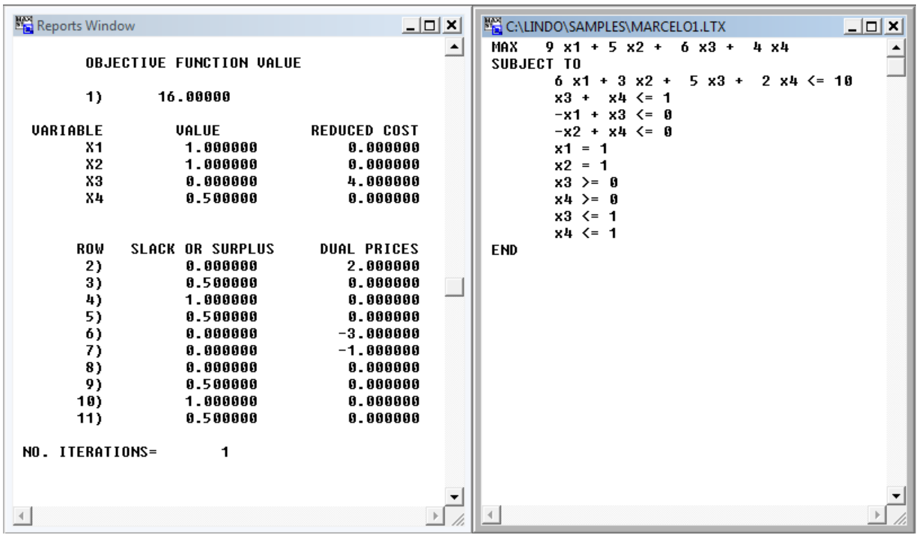


Partindo-se para 2ª iteração (usando-se x_2=0 e x_2=1), pelo nó x_1=1, para os sub-problemas relaxados, tem-se:

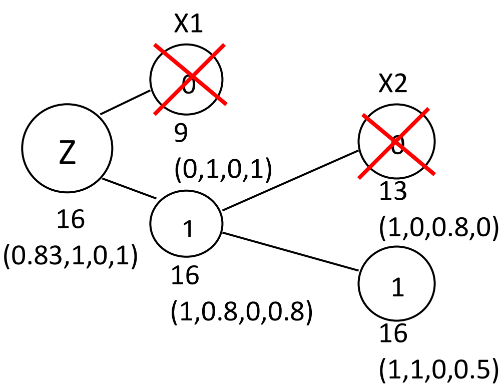
Para x_1=1 e x_2=0: x_3=0.8; x_4=0; Z=13,8 (Figura a seguir com a solução no software Lindo). Portanto Z ≤ 13 (valor inteiro).



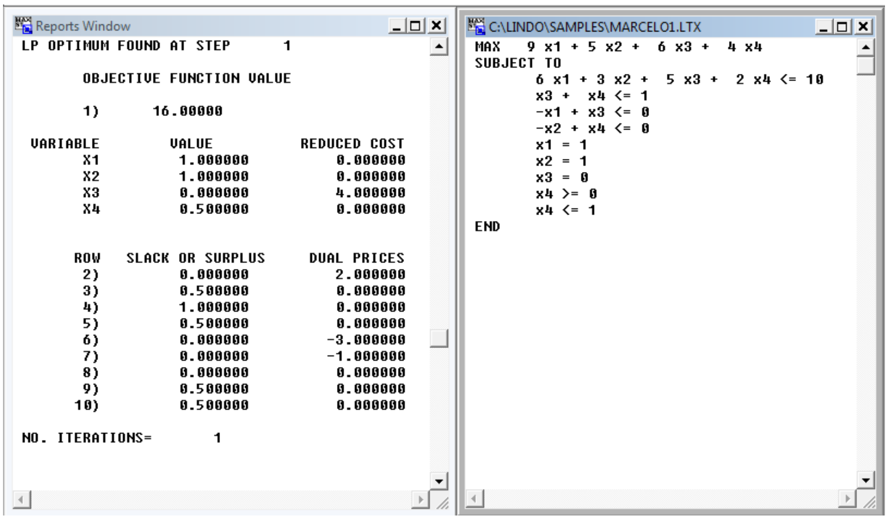
Para x_1=1 e x_2=1: x_3=0; x_4=0,5; Z=16 (Figura a seguir com a solução no software Lindo). Portanto Z ≤ 16.



A árvore montada a partir da fixação de x_1 e x_2 está exposta a seguir. A nova solução candidata é melhor que a anterior (Z*=13).



Para x_1=1, x_2=1 e **x_3=0**: x_4=0.5; Z=16 (Figura a seguir com a solução no software Lindo). Portanto Z ≤ 16.



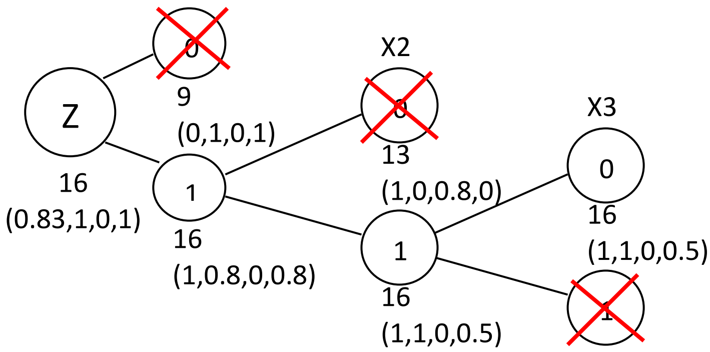
```
Reports Window
LP OPTIMUM FOUND AT STEP      1

       OBJECTIVE FUNCTION VALUE

       1)       16.00000

 VARIABLE        VALUE          REDUCED COST
       X1         1.000000          0.000000
       X2         1.000000          0.000000
       X3         0.000000          4.000000
       X4         0.500000          0.000000


      ROW   SLACK OR SURPLUS     DUAL PRICES
       2)         0.000000          2.000000
       3)         0.500000          0.000000
       4)         1.000000          0.000000
       5)         0.500000          0.000000
       6)         0.000000         -3.000000
       7)         0.000000         -1.000000
       8)         0.000000          0.000000
       9)         0.500000          0.000000
      10)         0.500000          0.000000

 NO. ITERATIONS=       1
```

```
C:\LINDO\SAMPLES\MARCELO1.LTX
MAX    9 x1 + 5 x2 +  6 x3 +  4 x4
SUBJECT TO
       6 x1 + 3 x2 +  5 x3 +  2 x4 <= 10
       x3 +  x4 <= 1
       -x1 + x3 <= 0
       -x2 + x4 <= 0
       x1 = 1
       x2 = 1
       x3 = 0
       x4 >= 0
       x4 <= 1
END
```

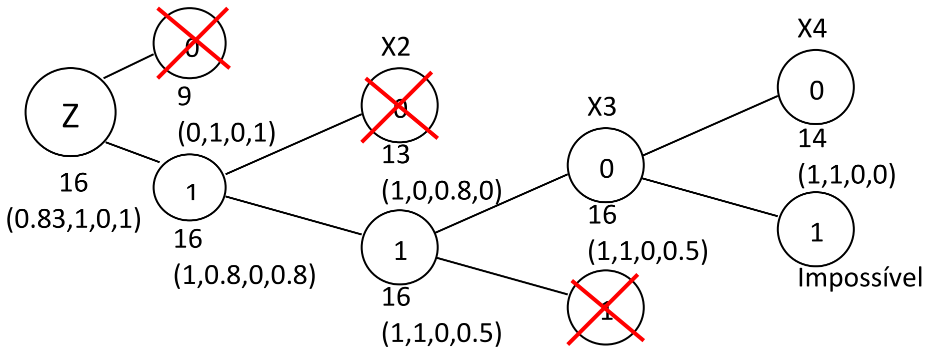
Para x_1=1, x_2=1 e **x_3=1** o modelo se apresenta sem soluções possíveis. O referido está exposto a seguir.

$$Max\ Z = 20 + 4x_4$$
$$sujeito\ a$$
$$2x_4 \leq -4$$
$$x_4 \leq 0$$
$$x_4 \leq 1$$
$$0 \leq x_j \ \leq 1$$

A árvore após a solução dos sub-problemas anteriores fica da seguinte forma:



Pela fixação dos valores de x_4 (x_4=0 e x_4=1) gera-se uma solução única, sem criar outros subproblemas. A árvore final fica:



Esta última solução candidata é melhor que a solução anterior (Z*=14). Como não há mais condições de criar sub-problemas, esta é a solução ótima.

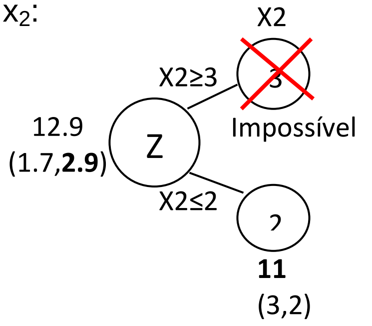
Resumidamente, para problemas de PLI de maximização, seguir os passos do método *Branch-and-Bound* a seguir:

1. Resolver o problema original usando programação linear, por exemplo, pelo método *Simplex*. Se a resposta satisfaz a restrição inteira, <u>esta é a solução ótima. Sendo assim, pare</u>, senão:
2. Encontrar uma solução viável que preencha a restrição inteira para uso como um limite superior. Usualmente para isso, arredonda-se a variável.
3. Ramificar pela variável de decisão do passo 1 que <u>não tenha um valor inteiro</u>. Caso todas as variáveis não sejam inteiras, iniciar a ramificação pela de maior valor do resíduo decimal. Dividir o problema em dois sub-problemas baseados nos valores inteiros que estão imediatamente abaixo ou acima do valor não inteiro. Esses limites deverão ser colocados na restrição do problema.
4. Criar nós no topo desses novos ramos pela solução dos novos problemas.
5. A) Se um ramo leva a uma <u>solução inviável</u> por programação linear, descarte o nó para continuidade da análise;
   B) Se um ramo leva a uma <u>solução viável</u> por programação linear, mas não é uma solução inteira vá para o passo 6;
   C) Se o ramo leva a uma solução inteira viável, examine o valor da função objetivo. Se este valor é igual ao limite inferior, uma solução ótima foi alcançada. Se ele não é igual ao limite inferior, mas ele é menor que o limite superior, adote-o como um novo limite superior e vá para o passo 6. Finalmente, se ele é maior que o limite superior, descarte esse ramo.
6. Examine ambos os ramos novamente e adote como limite superior o valor máximo da função objetivo para todos os nós finais. Se o limite inferior é igual ao limite superior, pare. Se não, volte ao passo 3.

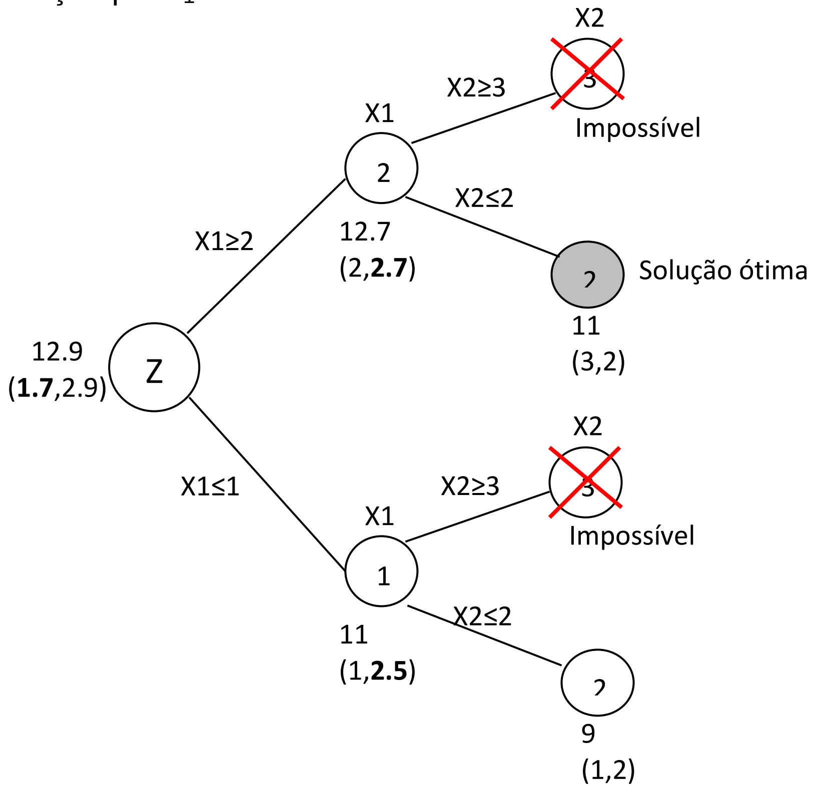
Exemplo 2 - Silva (s/d):

$Max\ Z = x_1 + 4x_2$
$sujeito\ a$
$-2x_1 + 4x_2 \leq 8$
$2x_1 + 3x_2 \leq 12$
$x_1, x_2 \geq 0$
$x_1, x_2\ são\ inteiros$

Iniciar a avaliação por x_2:



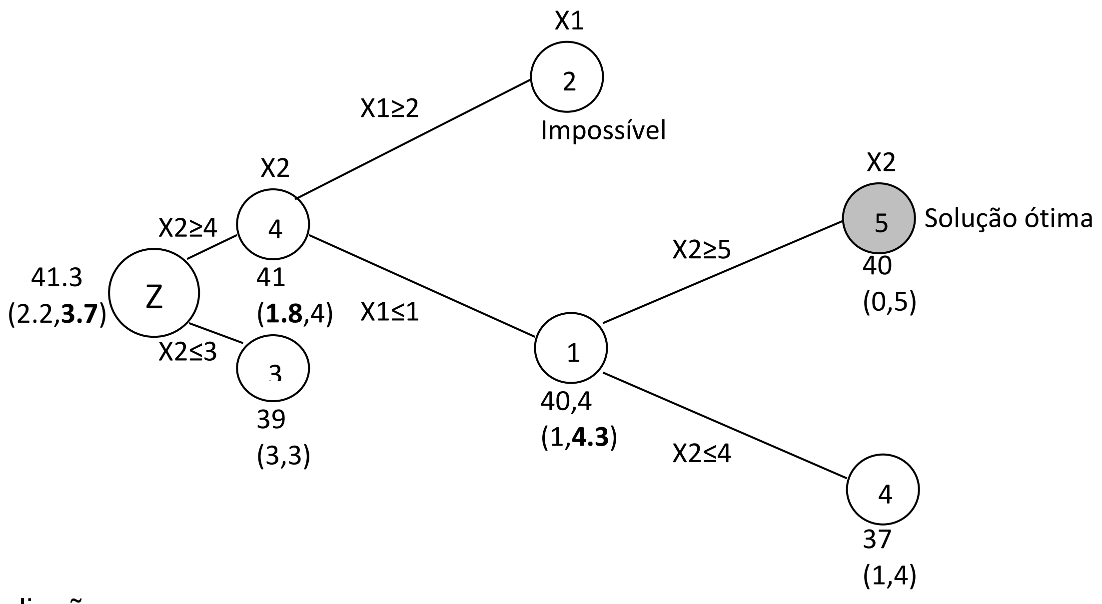
Avaliação por x_1:



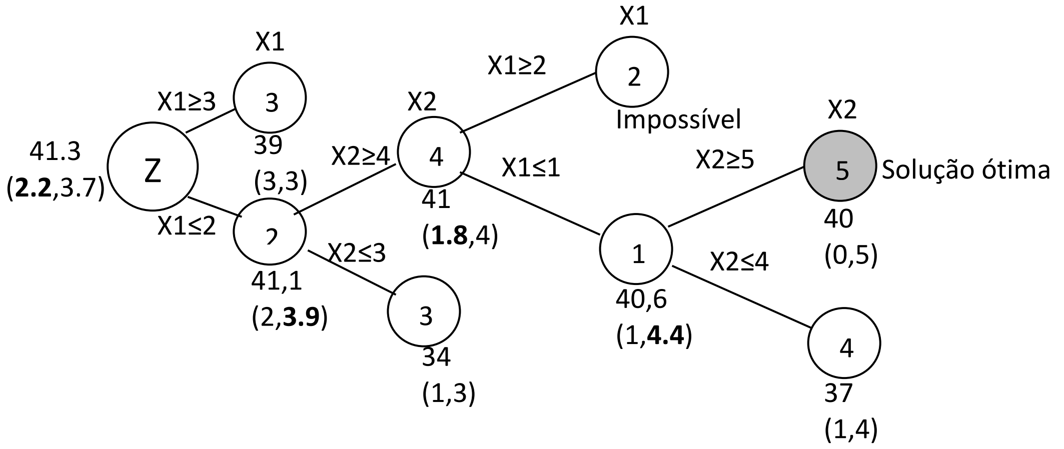
Exemplo 3 - Silva (s/d):

$$Max\ Z = 5x_1 + 8x_2$$
$$sujeito\ a$$
$$x_1 + x_2 \leq 6$$
$$5x_1 + 9x_2 \leq 45$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2\ são\ inteiros$$

Iniciar a avaliação por x_2:

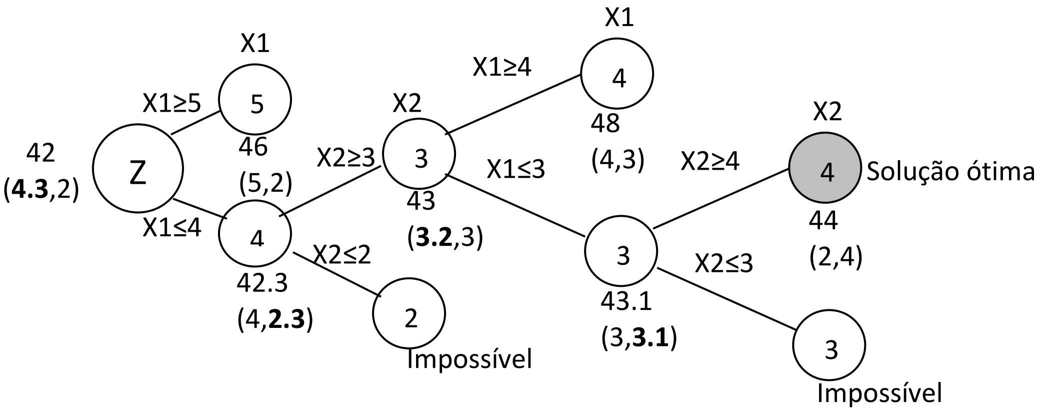


Avaliação por x_1:



Exemplo 4 - Silva (s/d):

$$Min\ Z = 6x_1 + 8x_2$$
$$sujeito\ a$$
$$6x_1 + 7x_2 \geq 40$$
$$x_2 \geq 2$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2\ são\ inteiros$$



7.4) PROGRAMAÇÃO DINÂMICA

A Programação Dinâmica procura resolver o problema de otimização por intermédio da análise de uma sequência de problemas mais simples do que o problema original.

A resolução do problema original de *n* variáveis é caracterizada pela determinação de uma variável e pela resolução de um problema que possua uma variável a menos (*n*-1). Este por sua vez é resolvido pela determinação de uma variável e pela resolução de um problema de *n*-2 variáveis e assim por diante.

O problema a ser resolvido é do tipo:

- Existem *n* atividades ou estágios numerados de 1 a *n*.
- X_i é a quantidade de recursos colocados nas atividades ou estágios *i* ($X_i \geq 0$)
- g_i (X_i) é a função que representa o ganho ou o retorno devido a colocação de X_i recursos na atividade *i*, $Q = X_1+X_2+...+X_n$ é a quantidade total de recursos disponíveis.
- O objetivo é determinar a distribuição de recursos X_i que maximiza o ganho total.
- $R(X_1, X_2,...,X_n) = g_1(X_1) + g_2(X_2) + ...g_n(X_n)$, considerando que as atividades são independentes e os ganhos g_i sejam aditivos.

Formulação

Maximizar R f(Q, n). Esta dependência é explicada da seguinte maneira:

$$f_n(Q) = \underset{X_i}{\text{Max}}\{\, R(X_1, X_2, \ldots, X_n)\}$$ onde:

$f_n(Q)$ representa o ganho máximo devido à distribuição de *Q* quantidades de recursos nas *n* atividades.

Condição Inicial

a) $g_i(0)=0$ ⇒ para cada atividade *i* (ganho nulo para zeros recursos distribuídos).

b) $f_n(0)=0$ ⇒ para n=1,2,... (se o total *Q* de recursos é nulo, o ganho máximo também é nulo).

c) $f_1(Q)=g_1(Q)$ ⇒ se existir n=1 atividade, então $R(X_1)=g_1(X_1)$.

Ao atribuir a quantidade X_n ($0 \le X_n \le Q$) de recursos à atividade *n*, restarão $Q-X_n$ recursos a serem distribuídos nas n-1 atividades restantes e o ganho máximo proveniente dessas n-1 atividades pode ser expresso por $f_{n-1}(Q-X_n)$. Sendo assim, o ganho total das *n* atividades pode ser expresso por $g_n(X_n)+f_{n-1}(Q-X_n)$. E se escolhermos X_n, que maximize esse ganho, tem-se o valor $f_n(Q)$ do ganho máximo devido à aplicação de Q recursos em *n* atividades. Tem-se então a relação fundamental da Programação Dinâmica, dada por $f_n(Q) = 0 \le \underset{X_n}{Max} \le Q \quad \{g_n(X_n) + f_{n-1}(Q - X_n)\}$

$para\ n = 2, 3, \ldots e;\ n = 1 \rightarrow f_1(Q) = g_1(Q)$

Exemplo 1:

Q = \$6,00 unidades de capital disponível;

n = 3 atividades diferentes para investimento e as funções de ganho $g_i(X_i)$ dadas pelo quadro abaixo:

Q	$g_1(Q)$	$g_2(Q)$	$g_3(Q)$
0	0	0	0
1	15	15	26
2	40	40	40
3	80	60	45
4	90	70	50
5	95	73	51
6	100	75	53

Qual a distribuição ótima do recurso Q = \$6,00 nas 3 atividades?

✓ Obtenção da função $f_1(Q)$ da atividade 1

Atividade n=1

Condição inicial:

$f_1(0) = g_1(0) = 0$	$f_1(2) = g_1(2) = 40$	$f_1(4) = g_1(4) = 90$	$f_1(6) = g_1(6) = 100$
$f_1(1) = g_1(1) = 15$	$f_1(3) = g_1(3) = 80$	$f_1(5) = g_1(5) = 95$	

✓ Obtenção da função $f_2(Q)$ da atividade 2

f_n (Q) para n=2

- Para Q=0, f_2 (0) = 0 pela condição inicial (b)
- Para Q=1, $f_2(1) = 0 \le \underset{X_2}{\text{Max}} \le 1 \quad \{g_2(X_2) + f_1(1 - X_2)\}$ e como os valores possíveis de X_2 são 0 e 1, tem-se:

  $$f_2(1) = \text{Max}\begin{Bmatrix} g_2(0) + f_1(1) = 0 + 15 = 15 \\ g_2(1) + f_1(0) = 15 + 0 = 15 \end{Bmatrix} = 15 \text{ para } X_2 = 0 \text{ ou } X_2 = 1$$

  escolhe-se, como solução ótima $X_2 = 0$ (poderia ter sido $X_2 = 1$).

- Para Q=2, $f_2(2) = 0 \le \underset{X_2}{\text{Max}} \le 2 \quad \{g_2(X_2) + f_1(2 - X_2)\}$ e como os valores possíveis de X_2 são 0, 1 e 2, tem-se:

  $$f_2(2) = \text{Max}\begin{Bmatrix} g_2(0) + f_1(2) = 0 + 40 = 40 \\ g_2(1) + f_1(1) = 15 + 15 = 30 \\ g_2(2) + f_1(0) = 40 + 0 = 40 \end{Bmatrix} = 40 \text{ para } X_2 = 0 \text{ ou } X_2 = 2$$

  escolhe-se, como solução ótima $X_2 = 0$ (poderia ter sido $X_2 = 2$).

- Para Q=3, $f_2(3) = 0 \le \underset{X_2}{\text{Max}} \le 3 \quad \{g_2(X_2) + f_1(3 - X_2)\}$ e como os valores possíveis de X_2 são 0, 1, 2 e 3 tem-se:

  $$f_2(3) = \text{Max}\begin{Bmatrix} g_2(0) + f_1(3) = 0 + 80 = 80 \\ g_2(1) + f_1(2) = 15 + 40 = 55 \\ g_2(2) + f_1(1) = 40 + 15 = 55 \\ g_2(3) + f_1(0) = 60 + 0 = 60 \end{Bmatrix} = 80 \text{ para } X_2 = 0$$

- Para Q=4, $f_2(4)=95$, para $X_2=1$
- Para Q=5, $f_2(5)=120$, para $X_2=2$
- Para Q=6, $f_2(6)=140$, para $X_2=3$

✓ Obtenção da função $f_3(Q)$ da atividade 3

De maneira análoga, obtém-se que f_3 (Q):

Para Q=2, $f_3(2) = 0 \leq \underset{X_3}{\text{Max}} \leq 2 \quad \{g_3(X_3) + f_2(2 - X_3)\}$

$$f_3(2) = \text{Max} \begin{Bmatrix} g_3(0) + f_2(2) = 0 + 40 = 40 \\ g_3(1) + f_2(1) = 26 + 15 = 41 \\ g_3(2) + f_2(0) = 40 + 0 = 40 \end{Bmatrix} = 41 \text{ para } X_3 = 1$$

Quadro dos Valores de f_N (Q):

Q	**X_1**	**$f_1(Q)$**	**X_2**	**$f_2(Q)$**	**X_3**	**$f_3(Q)$**
0	0	0	0	0	0	0
1	1	15	0	15	1	26
2	2	40	0	40	1	41
3	3	80	0	80	0	80
4	4	90	1	95	1	106
5	5	95	2	120	1	121
6	6	100	3	140	1	146

Ganho Máximo do Investimento:

Na coluna f_3 (Q) obtém-se como ganho máximo correspondente ao investimento nas 3 atividades, o valor \$146,00, para Q = 6.

A distribuição é:

a) para a atividade 3: X_3=1unidade alocada >> $f_3(Q)$=146 e subtraindo o ganho $g_3(1)$=26 (do quadro de ganhos) restam ainda 146-26=120 unidades que correspondem ao ganho da aplicação de Q=5 unidades nas outras 2 atividades.

b) para a atividade 2, o ganho de 120 unidades corresponde a aplicação de X_2=2 unidades na atividade e

c) para a atividade 1, restam, portanto, Q-X_3-X_2=3 unidades a serem aplicadas. Portanto, X_1=3.

Solução Ótima

X_1=3 com $g_1(3)$ = 80

X_2=2 com $g_2(2)$ = 40

X_3=1 com $g_3(1)$ = 26

e

R=g_1 + g_2 + g_3 = \$146,00

BIBLIOGRAFIA

Campos, Vânia B.G., **Otimização do Transporte**, Instituto Militar de Engenharia, Rio de Janeiro, 1998.

de Andrade, Eduardo Leopoldino, Introdução à Pesquisa Operacional, Editora LTC, ISBN 9788521616658, 4º Edição, Rio de Janeiro, 2009.

de Andrade, Eliana X.L.; Sampaio, Rubens e Silva, Geraldo N. **Notas em Matemática Aplicada** Sociedade Brasileira de Matemática Aplicada e Computacional, ISBN 85-7651-021-9, Editora SBMAC, São Carlos, 2005.

Goldbarg, Marco Cesar e Luna, Henrique Pacca L. **Otimização Combinatória e Programação Linear: Modelos e Algoritmos** Ed. Campus ISBN 8535215204, Rio de Janeiro, 2000.

Lachtermacher, Gerson **Pesquisa Operacional nas Tomadas de Decisões** Editora Campus, ISBN 8535220879, 1º Edição, Rio de Janeiro, 2006.

Novaes, Antônio Galvão, **Métodos de Otimização: aplicações aos transportes** Edgar Blucher, São Paulo, 1978

Silva, Ana Cristina Girão **Programação Linear Inteira *Branch-and-Bound***, Pesquisa Operacional II, Universidade Federal do Rio Grande do Norte – Departamento de Engenharia de Produção.

Silva, Arlindo **Programação Linear Inteira – Introdução**, Métodos de Apoio à Decisão, Departamento de Engenharia das Tecnologias da Informação, Instituto Politécnico de Castelo Branco, Escola Superior de Tecnologia, Portugal, 2001.

Smiderle, Andreia, **Técnicas da Pesquisa Operacional Aplicadas – Um Problema de Cobertura de Arcos**, Dissertação de Mestrado (Métodos Numéricos em Engenharia), Universidade Federal do Paraná, 153f., Curitiba, 2001.