OTIMIZAÇÃO COMBINATÓRIA

Marcone Jamilson Freitas Souza

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Homepage: http://www.decom.ufop.br/prof/marcone

E-mail: marcone.freitas@yahoo.com.br

Sumário

Parte I
Programação Inteira

1 Introdução

1.1 Características dos modelos lineares de programação inteira

[Retirado de Goldbarg e Luna (2005), vide [3]] Uma confeitaria pode produzir dois tipos de sorvete em lata: chocolate e creme. Cada lata do sorvete de chocolate é vendido com um lucro de \$3 e as latas de creme com um lucro de \$1. Contratos com várias lojas impõem que sejam produzidas no mínimo 10 latas de sorvete de chocolate por dia e que o total de latas fabricadas por dia nunca seja menor que 20. O mercado só é capaz de consumir até 40 latas de sorvete de creme e 60 de chocolate. As máquinas de preparação do sorvete disponibilizam 180 horas de operação por dia, sendo que cada lata de sorvete de chocolate consome 2 horas de trabalho e cada lata de creme, 3 horas. Determine o esquema de produção que maximiza os lucros com a venda de latas de sorvete.

Solução:

(a) Variáveis de decisão:
x_j = número de latas de sorvete do tipo j a serem produzidas por dia, sendo $j = 1$ (chocolate) e $j = 2$ (creme)

(b) Função objetivo:

$$\max f(x) = \underbrace{\frac{3}{\$/(\text{lata de chocolate})}}_{} \times \underbrace{x_1}_{(\text{latas de chocolate})} + \underbrace{\frac{1}{\$/(\text{lata de creme})}}_{} \times \underbrace{x_2}_{(\text{latas de creme})}$$

(c) Restrições:

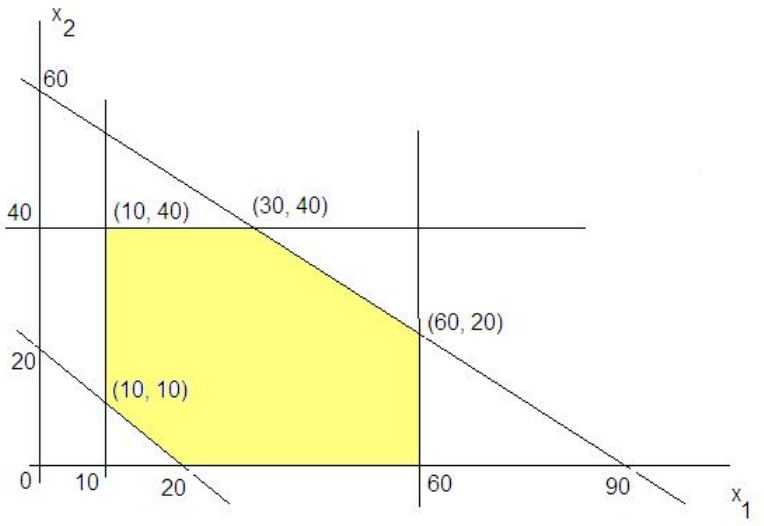
c.1) Demanda do mercado:
$x_1 \leq 60$ (O mercado não absorve mais do que 60 latas de sorvete de chocolate por dia)
$x_2 \leq 40$ (Não há demanda para mais do que 40 latas de sorvete de creme por dia)

c.2) Contrato com as lojas:
$x_1 \geq 10$ (Exige-se uma produção mínima diária de 10 latas de sorvete de chocolate)
$x_1 + x_2 \geq 20$ (Exige-se uma produção mínima diária de 20 latas de sorvete)

c.3) Disponibilidade das máquinas:
$2x_1 + 3x_2 \leq 180$ (Há apenas 180 horas de operação disponíveis nas máquinas por dia)

c.4) Integralidade e não-negatividade:
$x_1, x_2 \in \mathbb{Z}^+$

Para resolver esse Problema de Programação Linear Inteira (PLI), construamos graficamente a região viável:



Propriedade de um PPL: O ótimo de um problema de programação linear (PPL), se existir, estará em um vértice do politopo definido pela região viável.

Do Cálculo Diferencial e Integral sabemos que a direção e sentido de máximo crescimento de uma função f é determinada pelo seu gradiente $\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \cdots, \frac{\partial f(x)}{\partial x_n}\right)$.

Sendo a função $f(x) = 3x_1 + x_2$, resulta que $\nabla f(x) = (3, 1)$. Para conhecermos a direção e sentido do gradiente, da origem caminhamos 3 unidades no sentido Ox_1 e uma unidade no sentido Ox_2. Pela regra de composição de forças (regra do paralelograma), tem-se o sentido e direção do gradiente. Traça-se, a seguir, uma reta perpendicular à direção do gradiente. Caminhando-se com essa reta no sentido apontado pelo gradiente até tangenciar a região viável tem-se, no ponto de tangência, a solução ótima. Logo, o ótimo do PPL em questão ocorre no vértice de coordenadas $(60, 20)$, com valor ótimo dado por $f(x^*) = 200$, isto é, no esquema ótimo de produção, devem ser produzidas diariamente 60 latas de sorvete de chocolate e 20 de creme, com um retorno de \$ 200.

No exemplo dado, a solução ótima ocorreu em um ponto de coordenadas inteiras. No entanto, nem sempre isso ocorre, conforme mostra o exemplo a seguir:

$$\begin{aligned} \max f(x) = {} & x_1 + 19x_2 \\ & x_1 + 20x_2 \leq 50 \\ & x_1 + \quad x_2 \leq 20 \\ & x_1, \ \ x_2 \ \in \ \mathbb{Z}^+ \end{aligned}$$

Fazendo-se a relaxação linear desse problema, isto é, desprezando-se as restrições de integralidade das variáveis e assumindo que as mesmas são não-negativas ($x_1 \geq 0$ e $x_2 \geq 0$), a solução ótima desse PPI relaxado é:

$$\begin{aligned} x_1^* &= 18,89 \\ x_2^* &= 1,58 \\ f(x^*) &= 48,42 \end{aligned}$$

Vamos supor que seja uma boa estratégia determinar os inteiros mais próximos do entorno dessa solução contínua. Vejamos os valores da função objetivo para as possíveis combinações:

x_1	x_1	$f(x)$
19	2	inviável
18	1	37
19	1	38
18	2	inviável

Se essa estratégia fosse correta, concluiríamos, de forma EQUIVOCADA, que $x_1 = 19$ e $x_2 = 1$ é a solução ótima do problema, com valor ótimo dado por $f(x) = 38$. Na realidade, a solução ótima é: $x_1^* = 10$ e $x_2^* = 2$, com $f(x^*) = 48$, ou seja, o valor ótimo difere em 21% do anteriormente apontado.

Esse exemplo mostra que pode não ser uma boa estratégia arredondar os valores do entorno de uma solução ótima contínua como uma forma de determinar a solução ótima do problema de variáveis inteiras. Mais à frente (Seção 3) mostraremos a técnica *branch-and-bound*, destinada a encontrar a solução ótima de um problema de programação inteira.

2 Modelagem de Programação Matemática Inteira

2.1 Alocação de recursos

[Retirado de [4]] A Capitão Caverna S.A., localizada em Pedra Lascada, aluga 3 tipos de barcos para passeios marítimos: jangadas, supercanoas e arcas com cabine. A companhia fornece juntamente com o barco um capitão para navegá-lo e uma tripulação que varia de acordo com a embarcação: uma para jangadas, duas para supercanoas e três para arcas. A companhia tem 4 jangadas, 8 supercanoas e 3 arcas e em seu corpo de funcionários: 10 capitães e 18 tripulantes. O aluguel é por diárias e a Capitão Caverna lucra \$50 por jangada, \$70 por supercanoa e \$100 por arca. Faça um modelo de programação matemática que determine o esquema de aluguel que maximiza o lucro.

Solução:

(a) Variáveis de decisão:
x_i = número de embarcações do tipo i a serem alugadas, sendo $i = 1$ (jangada), $i = 2$ (supercanoa) e $i = 3$ (arca com cabine).

(b) Função objetivo:
$\max f(x) = 50x_1 + 70x_2 + 100x_3$

(c) Restrições:

c.1) Número de capitães:
$x_1 + x_2 + x_3 \leq 10$ (Há somente 10 capitães)

c.2) Número de tripulantes:
$x_1 + 2x_2 + 3x_3 \leq 18$ (Há somente 18 tripulantes)

c.3) Quantidade de jangadas:
$x_1 \leq 4$ (O número de jangadas está limitado a 4)

c.4) Quantidade de supercanoas:
$x_2 \leq 8$ (Há apenas 8 supercanoas)

c.5) Quantidade de arcas com cabine:
$x_3 \leq 3$ (Há apenas 3 arcas com cabine disponíveis)

c.6) Integralidade e não-negatividade:
$x_1, x_2, x_3 \in \mathbb{Z}^+$

Para fazer um modelo genérico desse PPL, coloquemos os dados em uma tabela e façamos as seguintes convenções:

emb	:	Conjunto dos diferentes tipos de embarcação = {Jangada, Supercanoa, Arca}
l_i	:	Lucro proporcionado pelo aluguel da embarcação do tipo i
cap_i	:	Número de capitães necessários para comandar a embarcação do tipo i
$trip_i$	:	Número de tripulantes necessários para trabalhar na embarcação do tipo i
$disp_i$	:	Número disponível de embarcações do tipo i
$ntrips$	:	Número de tripulações disponíveis
$ncapitaes$	:	Número de capitães disponíveis

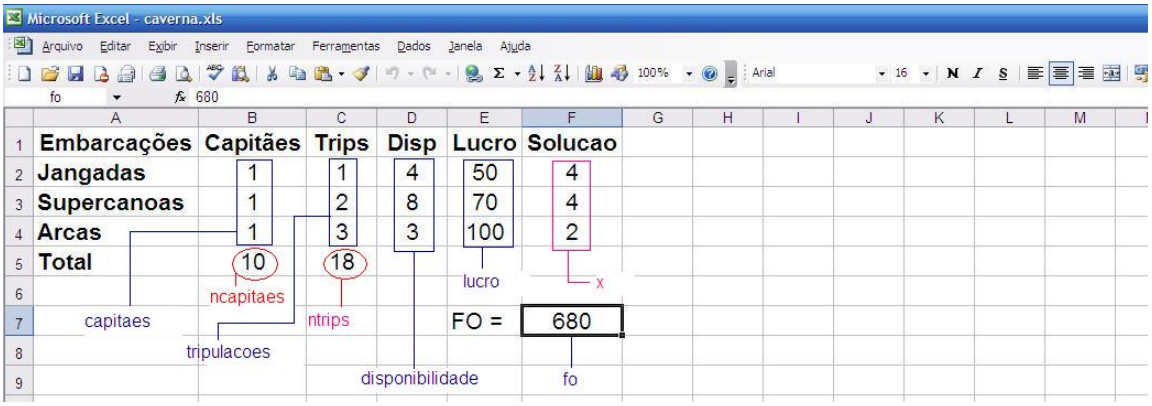
Tipo de embarcação	# Tripulantes req.	# Capitães req.	# Emb. disp.	Lucro ($)
Jangada	1	1	4	50
Supercanoa	2	1	8	70
Arca	3	1	3	100
Funcionários disp.	18	10		

O modelo genérico relativo ao problema em questão pode ser assim formulado:

$$
\begin{array}{lrcll}
\max & \displaystyle\sum_{i \in emb} l_i x_i & & & \\
& \displaystyle\sum_{i \in emb} cap_i x_i & \leq & ncapitaes & \\
& \displaystyle\sum_{i \in emb} trip_i x_i & \leq & ntrips & \\
& x_i & \leq & disp_i & \forall i \in emb \\
& x_i & \in & \mathbb{Z}^+ & \forall i \in emb
\end{array}
$$

Segue uma implementação LINGO interfaceando com um arquivo Excel, onde se considera a seguinte correspondência de nomes para os blocos de células:

Bloco de células	Nome
A2:A4	embarcacoes
B2:B4	capitaes
C2:C4	tripulacoes
D2:D4	disponibilidade
E2:E4	lucro
B5	ncapitaes
C5	ntrips
F2:F4	x
F7	fo




```
sets:
  emb /@ole('caverna.xls','embarcacoes')/:l, x, cap, trip, disp;
endsets

data:
  l = @ole('caverna.xls','lucro');
  cap = @ole('caverna.xls','capitaes');
  trip = @ole('caverna.xls','tripulacoes');
  disp = @ole('caverna.xls','disponibilidade');
  ncapitaes = @ole('caverna.xls','ncapitaes');
  ntrips = @ole('caverna.xls','ntrips');
enddata

[fo] max = @sum(emb(i): l(i)*x(i));

[fcap] @sum(emb(i): cap(i)*x(i)) <= ncapitaes;

[ftrip] @sum(emb(i): trip(i)*x(i)) <= ntrips;

@for(emb(i): [fdisp] x(i) <= disp(i));

@for(emb(i): @gin(x(i)));

data:
  @ole('caverna.xls','x') = x;
  @ole('caverna.xls','fo') = fo;
enddata
```

A solução ótima para o problema é alugar 4 jangadas, 4 supercanoas e 2 arcas, produzindo um lucro máximo de R$680,00.

2.2 Problema da Mochila 0-1 (*Knapsack Problem*)

Um excursionista planeja fazer uma viagem acampando. Há 5 itens que ele deseja levar consigo, mas estes, juntos, excedem o limite de 60 quilos que ele supõe ser capaz de carregar. Para ajudar a si próprio no processo de seleção, ele atribui valores, por ordem crescente de importância a cada um dos itens conforme a tabela a seguir:

Item	1	2	3	4	5
Peso (Kg)	52	23	35	15	7
Valor	100	60	70	15	8

Supondo a existência de uma unidade de cada item, faça um modelo de programação inteira que maximize o valor total sem exceder as restrições de peso.

Solução:

(a) Variáveis de decisão:

$$x_j = \begin{cases} 1, & \text{se o item } j \text{ for colocado na mochila;} \\ 0, & \text{caso contrário.} \end{cases}$$

(b) Função objetivo:
$\max f(x) = 100x_1 + 60x_2 + 70x_3 + 15x_4 + 8x_5$

(c) Restrições:

c.1) Limite de peso:
$52x_1 + 23x_2 + 35x_3 + 15x_4 + 7x_5 \leq 60$ (Pode-se carregar 60 Kg, no máximo)

c.2) Integralidade:
$x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}$

Considerando a notação a seguir:

$itens$	:	Conjunto dos itens = {1, 2, 3, 4, 5 }
cap	:	Capacidade da mochila
w_j	:	Peso relativo ao item j
p_j	:	Valor de retorno proporcionado pelo item j

o problema da mochila 0-1 pode ser formulado como:

$$\begin{array}{lrcll} \max & \sum\limits_{j \in itens} p_j x_j & & & \\ & \sum\limits_{j \in itens} w_j x_j & \leq & cap & \\ & x_j & \in & \{0,1\} & \forall j \in itens \end{array}$$

2.3 Problema da Mochila Inteira

Trata-se de uma extensão do problema anterior, na qual para cada item j existem u_j unidades disponíveis. A modelagem de programação inteira deste problema é:

$$\begin{array}{lrcll} \max & \sum\limits_{j \in itens} p_j x_j & & & \\ & \sum\limits_{j \in itens} w_j x_j & \leq & cap & \\ & x_j & \leq & u_j & \forall j \in itens \\ & x_j & \in & \mathbb{Z}^+ & \forall j \in itens \end{array}$$

em que a variável de decisão x_j indica o número de unidades do item j alocados à mochila.

2.4 Problema da Mochila 0-1 Múltipla

Neste problema, além do conjunto de itens, cada qual com peso w_j e valor de retorno p_j, há um conjunto de mochilas, cada qual com capacidade cap_i. Existe uma unidade de cada item e o objetivo é também maximizar o valor de retorno dos itens alocados às mochilas.

A modelagem de programação inteira deste problema é:

$$\begin{array}{lrcll} \max & \sum_{i \in mochilas} \sum_{j \in itens} p_j x_{ij} & & & \\ & \sum_{j \in itens} w_j x_{ij} & \leq & cap_i & \forall i \in mochilas \\ & \sum_{i \in mochilas} x_{ij} & \leq & 1 & \forall j \in itens \\ & x_{ij} & \in & \{0,1\} & \forall i \in mochilas\ , \forall j \in itens \end{array}$$

em que a variável de decisão x_{ij} assume valor 1 se o item j for alocado à mochila i e 0, caso contrário. O primeiro conjunto de restrições assegura que cada mochila i não comporta mais que cap_i unidades de peso, enquanto o segundo conjunto impede que um mesmo item j seja alocado a mais de uma mochila.

2.5 Problema da Mochila Inteira Múltipla

Este problema difere do anterior no sentido de que neste problema pode haver mais de uma unidade de cada item; no caso, há u_j unidades disponíveis de cada item j.

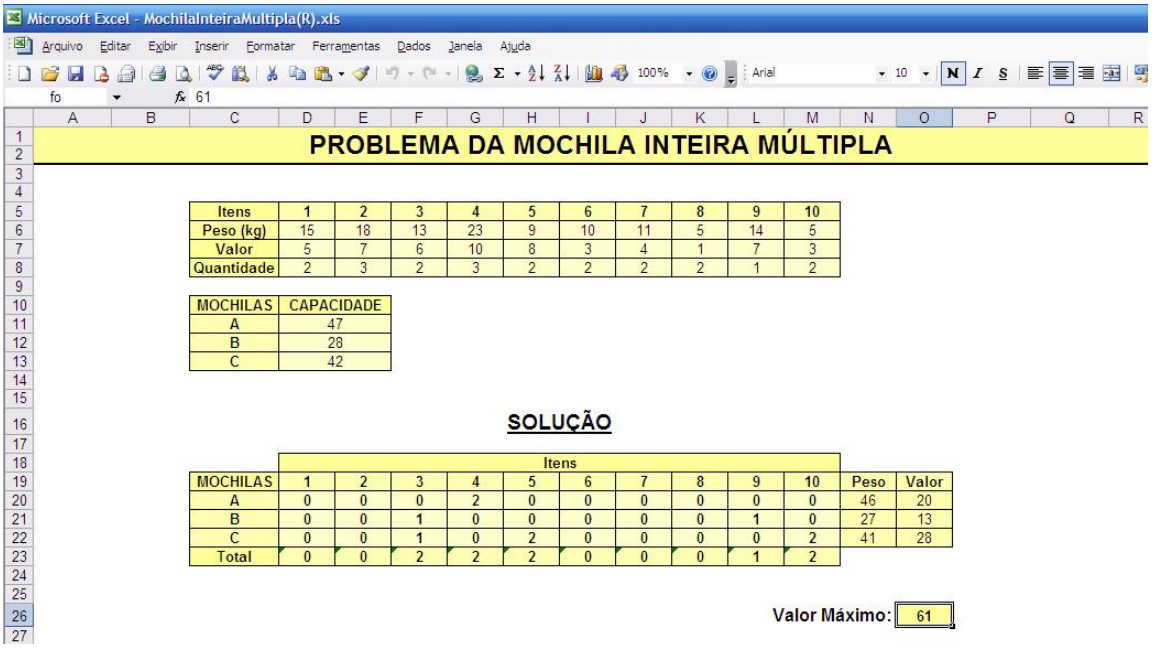
A modelagem de programação inteira deste problema é:

$$\begin{array}{lrcll} \max & \sum_{i \in mochilas} \sum_{j \in itens} p_j x_{ij} & & & \\ & \sum_{j \in itens} w_j x_{ij} & \leq & cap_i & \forall i \in mochilas \\ & \sum_{i \in mochilas} x_{ij} & \leq & u_j & \forall j \in itens \\ & x_{ij} & \in & \mathbb{Z}^+ & \forall i \in mochilas\ , \forall j \in itens \end{array}$$

Neste modelo, x_{ij} indica a quantidade de itens j alocados à mochila i. Mostra-se, a seguir, uma implementação LINGO deste problema.

No arquivo Excel considerado, há a seguinte correspondência de nomes para os blocos de células:

Bloco de células	Nome
D5:M5	Itens
C11:C13	Mochilas
D6:M6	peso
D7:M7	beneficio
D11:D13	capacidade
D8:M8	u
D20:M22	x
O26	fo



Itens	1	2	3	4	5	6	7	8	9	10
Peso (kg)	15	18	13	23	9	10	11	5	14	5
Valor	5	7	6	10	8	3	4	1	7	3
Quantidade	2	3	2	3	2	2	2	2	1	2

MOCHILAS	CAPACIDADE
A	47
B	28
C	42

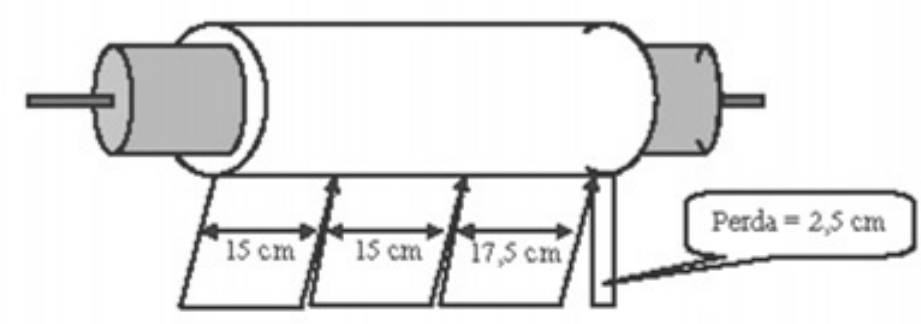
	Itens											
MOCHILAS	1	2	3	4	5	6	7	8	9	10	Peso	Valor
A	0	0	0	2	0	0	0	0	0	0	46	20
B	0	0	1	0	0	0	0	0	1	0	27	13
C	0	0	1	0	2	0	0	0	0	2	41	28
Total	0	0	2	2	2	0	0	0	1	2		

```
sets:
  Itens /@ole('MochilaInteiraMultipla(R).xls','Itens')/: w, p, u;
  Mochilas/@ole('MochilaInteiraMultipla(R).xls','mochilas')/: cap;
  matriz(Mochilas, Itens): x;
endsets

data:
  w, p, cap, u = @ole('MochilaInteiraMultipla(R).xls',
                      'peso','beneficio','capacidade','u');
enddata

! Maximizar o benefício pelo uso dos Itens;
[fo] max = @sum(Mochilas(i): @sum(Itens(j): p(j)*x(i,j)));

! A capacidade da mochila não pode ser superada;
@for(Mochilas(i): @sum(Itens(j): w(j)*x(i,j)) <= cap(i));

! Existem uj unidades de cada item j;
@for(Itens(j): @sum(Mochilas(i): x(i,j)) <= u(j));

! É permitido levar um número inteiro de Itens;
@for(Mochilas(i): @for(Itens(j): @gin(x(i,j))));

data:
  @ole('MochilaInteiraMultipla(R).xls','x','fo') = x, fo;
enddata
```

Na solução ótima deste problema, a mochila **A** recebe 2 unidades do item 4; a mochila **B** recebe uma unidade do item 3 e uma do item 9, enquanto a mochila **C** recebe uma unidade do item 3, duas unidades do item 5 e duas do item 10. O valor de retorno máximo é 61.

2.6 Problema de Corte de Estoque (*Cutting Stock Problem*)

Certa empresa trabalha com a produção de etiquetas autocolantes. O papel usado para sua confecção encontra-se em bobinas de mesmo comprimento, todas com largura de 50 cm. As encomendas para a próxima semana impõem a necessidade de se cortarem 32 bobinas de 15 cm de largura, 17 bobinas de 17,5 cm de largura e 21 bobinas de 20 cm de largura. É política da empresa manter em estoque o excedente ao pedido em quantidade máxima de 10 bobinas cortadas de acordo com a encomenda. Esta ação evita a imobilização de capital, uma vez que são incertos os próximos pedidos.



O departamento técnico relacionou na tabela abaixo as possíveis programações de cortes, tendo em vista as encomendas.

Programações de corte	**Largura da Faixa Cortada**			**Desperdício (cm)**
	15cm	**17,5cm**	**20cm**	
1	3	0	0	5
2	2	1	0	2,5
3	1	2	0	0
4	2	0	1	0
5	0	1	1	12,5
6	0	0	2	10

Elabore um modelo de programação inteira que determine a estratégia a ser seguida pela empresa de forma a minimizar os desperdícios face à necessidade de produção.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$padroes$: Conjunto de padrões de corte;
$bobinas$: Conjunto dos tipos de bobinas;
$desp_i$: Desperdício relativo aos cortes realizados de acordo com o padrão i;
$demanda_j$: Demanda por bobinas do tipo j;
a_{ij} : Quantidade de bobinas do tipo j produzidas no padrão de corte i;
$estmax$: Estoque máximo de bobinas permitido;

e as seguintes variáveis de decisão:

x_i : Número de cortes realizados segundo o padrão i

O modelo de programação matemática para esse problema de corte de estoque é:

$$
\begin{aligned}
\min \quad & \sum_{i \in padroes} desp_i x_i & & & \\
& \sum_{i \in padroes} a_{ij} x_i & \geq & demanda_j & \forall j \in bobinas \\
& \sum_{i \in padroes} a_{ij} x_i & \leq & demanda_j + estmax & \forall j \in bobinas \\
& x_i & \in & \mathbb{Z}^+ & \forall i \in padroes
\end{aligned}
$$

Nesta formulação, o objetivo é minimizar as perdas com os padrões de corte. O primeiro conjunto de restrições assegura o atendimento da demanda por bobinas do tipo j, enquanto o segundo indica que para cada tipo de bobina j, não podem ser estocadas mais que $estmax$ unidades.

2.7 Problema de Corte de Estoque Unidimensional

Uma serralheria dispõe de barras de 7 metros de comprimento que devem ser cortadas para obter barras menores atendendo a uma encomenda. As seguintes quantidades e tamanhos são requeridos: 92 barras de 2 metros, 59 barras de 3 metros e 89 barras de 4 metros. Elabore um modelo de programação linear inteira que minimize as perdas com os cortes.

Solução:
Nesse problema não são dados os padrões de corte, sendo necessário determiná-los previamente. A tabela a seguir relaciona os possíveis padrões de corte formados a partir do corte de barras de 7 metros de comprimento, bem como as perdas relativas a cada padrão.

	Quant. de barras por tipo			Perda
Padrão	2 m.	3 m.	4 m.	(m.)
1	0	1	1	0
2	2	1	0	0
3	3	0	0	1
4	0	2	0	1
5	1	0	1	1

Modelo de Programação Matemática:

Sejam os seguintes dados de entrada para o problema:

$padroes$: Conjunto dos possíveis padrões de corte;
$barras$: Conjunto de barras;
$perda_i$: Perda com os cortes realizados de acordo com o padrão i;
$demanda_j$: Demanda por barras do tipo j;
a_{ij} : Quantidade de barras do tipo j produzidas no padrão de corte i;

e as seguintes variáveis de decisão:

x_i : Número de cortes realizados segundo o padrão i

Assim, o modelo de programação matemática é:

$$\begin{aligned}
\min \quad & \sum_{i \in padroes} perda_i x_i \\
& \sum_{i \in padroes} a_{ij} x_i \geq demanda_j \qquad \forall j \in barras \\
& x_i \in \mathbb{Z}^+ \qquad \forall i \in padroes
\end{aligned}$$

Observação:
Relativamente ao problema anterior, considere que a serralheria não tem espaço para reaproveitar as barras menores não usadas. Elabore um modelo de programação linear inteira que minimize as perdas com os cortes e com o excesso de barras menores não aproveitadas.

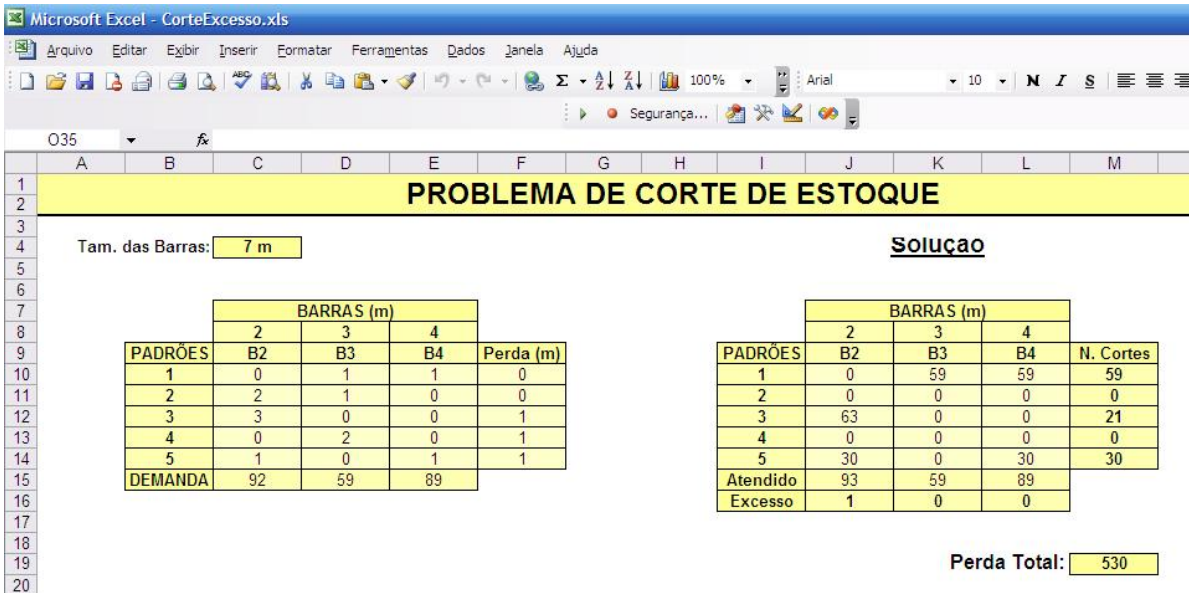
Nessa nova situação, chamando de $dimenbarra_j$ o comprimento da barra do tipo j, em metros, então o modelo de programação matemática que contempla esse novo objetivo é:

$$\begin{aligned}
\min \quad & \sum_{i \in padroes} perda_i x_i + \sum_{j \in barras} dimenbarra_j \times \left(\sum_{i \in padroes} a_{ij} x_i - demanda_j \right) \\
& \sum_{i \in padroes} a_{ij} x_i \geq demanda_j \qquad \forall j \in barras \\
& x_i \in \mathbb{Z}^+ \qquad \forall i \in padroes
\end{aligned}$$

Nesta nova função objetivo, a componente $\left(\sum_{i \in padroes} a_{ij} x_i - demanda_j \right)$ indica o excesso de barras do tipo j geradas. Multiplicando-a por $dimenbarra_j$ tem-se o valor da perda, em metros, devido ao excesso de barras produzidas.

A seguir, uma implementação LINGO do problema interfaceando com um arquivo Excel. No arquivo Excel considerado, há a seguinte correspondência de nomes para os blocos de células:

Bloco de células	Nome
B10:B14	padroes
C9:E9	barras
C8:E8	dimenbarra
F10:F14	perda
C10:E14	a
C15:E15	demanda
J16:L16	excesso
M10:M14	solucao
M19	ptotal



Microsoft Excel - CorteExcesso.xls

Arquivo Editar Exibir Inserir Formatar Ferramentas Dados Janela Ajuda

O35

PROBLEMA DE CORTE DE ESTOQUE

Tam. das Barras: 7 m

	BARRAS (m)			
	2	3	4	
PADRÕES	B2	B3	B4	Perda (m)
1	0	1	1	0
2	2	1	0	0
3	3	0	0	1
4	0	2	0	1
5	1	0	1	1
DEMANDA	92	59	89	

Soluçao

	BARRAS (m)			
	2	3	4	
PADRÕES	B2	B3	B4	N. Cortes
1	0	59	59	59
2	0	0	0	0
3	63	0	0	21
4	0	0	0	0
5	30	0	30	30
Atendido	93	59	89	
Excesso	1	0	0	

Perda Total: 530

```
sets:
  padroes/@ole('CorteExcesso.xls','padroes')/: perda, x;
  barras/@ole('CorteExcesso.xls','barras')/: demanda, dimenbarra;
  matriz(padroes,barras): a;
endsets

data:
  perda      =  @ole('CorteExcesso.xls','perda');
  demanda    =  @ole('CorteExcesso.xls','demanda');
  a          =  @ole('CorteExcesso.xls','a');
  dimenbarra =  @ole('CorteExcesso.xls','dimenbarra');
enddata

[fo] min = @sum(padroes(i): perda(i)*x(i)) +
           (@sum(barras(j):
              dimenbarra(j) * @sum(padroes(i): a(i,j)*x(i)) - demanda(j)));

@for(barras(j): [excbarra] @sum(padroes(i): a(i,j)*x(i)) >= demanda(j));

@for(padroes(i): @GIN(x(i)));

data:
  @ole('CorteExcesso.xls','excesso','solucao','ptotal') = excbarra, x, fo;
enddata
```

Como se observa, na solução ótima devem ser cortadas 59 peças no padrão 1, 21 no padrão 3 e 30 no padrão 5. A perda total é de 530 metros, sendo que apenas uma barra de 2 metros foi cortada em excesso.

2.8 Alocação de pessoal (*Staff Scheduling*)

Um hospital trabalha com atendimento variável em demanda durante as 24 horas do dia. As necessidades distribuem-se segundo a tabela:

Turno	Horário	Número requerido de enfermeiros
1	08 às 12 h	51
2	12 às 16 h	58
3	16 às 20 h	62
4	20 às 24 h	41
5	24 às 04 h	32
6	04 às 08 h	19

O horário de trabalho de um enfermeiro é de 8 horas seguidas e só pode ser iniciado no começo de cada turno, isto é, às 8 ou 12 ou 16 ou 20 ou 24 ou 04 horas. Elabore um modelo de PLI que minimize o gasto com a mão-de-obra. Considere que cada enfermeiro recebe \$100 por hora de trabalho no período diurno (08 às 20 h) e \$125 no período noturno (20 às 08 h).

Solução:

(a) Variáveis de decisão:
x_i = número de enfermeiros que iniciam sua jornada no início do turno i.

(b) Função objetivo:
$\min f(x) = 800x_1 + 800x_2 + 900x_3 + 1000x_4 + 1000x_5 + 900x_6$

(c) Restrições:

c.1) Número de enfermeiros necessários no turno 1:
$x_1 + x_6 \geq 51$

c.2) Número de enfermeiros necessários no turno 2:
$x_2 + x_1 \geq 58$

c.3) Número de enfermeiros necessários no turno 3:
$x_3 + x_2 \geq 62$

c.4) Número de enfermeiros necessários no turno 4:
$x_4 + x_3 \geq 41$

c.5) Número de enfermeiros necessários no turno 5:
$x_5 + x_4 \geq 32$

c.6) Número de enfermeiros necessários no turno 6:
$x_6 + x_5 \geq 19$

c.7) Integralidade e não-negatividade:
$x_1, x_2, x_3, x_4, x_5, x_6 \in \mathbb{Z}^+$

Para fazer um modelo genérico deste PPLI, utilizemos as seguintes notações:

$turnos$: Conjunto dos turnos de trabalho = {1, 2, 3, 4, 5, 6 }
c_i : Custo do enfermeiro que inicia sua jornada no turno i
$demanda_i$: Número de enfermeiros necessários durante o turno i

O modelo genérico relativo ao problema em questão pode ser assim formulado:

$$\min \sum_{i \in turnos} c_i x_i$$
$$x_i + x_{i-1} \geq demanda_i \quad \forall i \in turnos$$
$$x_i \in \mathbb{Z}^+ \quad \forall i \in turnos$$

As restrições $x_i + x_{i-1} \geq demanda_i$ indicam que em cada turno i trabalham os enfermeiros que iniciaram sua jornada no turno i, bem como aqueles que começaram sua jornada de trabalho no turno anterior.

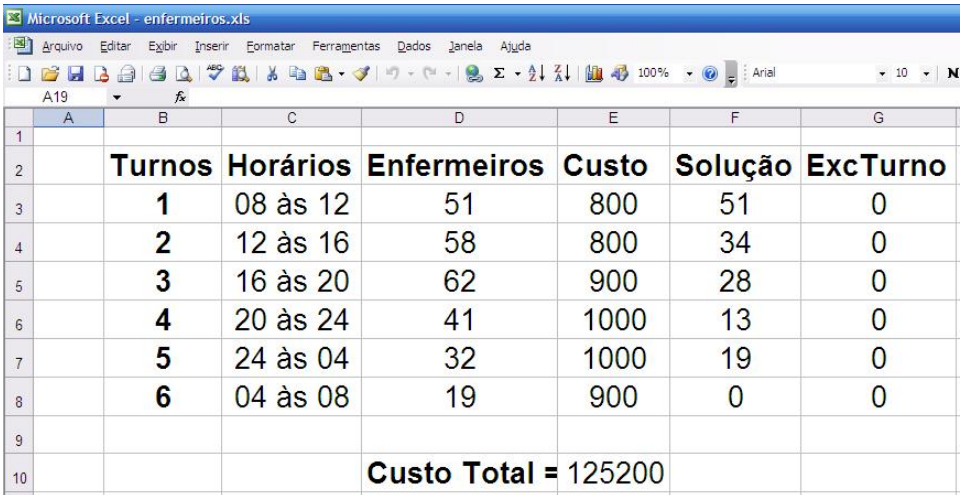
Em princípio, essas restrições não podem ser utilizadas diretamente, uma vez que quando $i = 1$, a variável x_{i-1} não está definida. Uma solução seria substituí-las pelas duas restrições seguintes: $x_i + x_{i-1} \geq demanda_i \; \forall i \in turnos \;\mid\; i \neq 1$ e $x_1 + x_{|turnos|} \geq demanda_1$, onde $|turnos|$ representa a cardinalidade do conjunto *turnos*.

No LINGO, uma solução para esta situação é usar uma lista circular. A função que faz isso no LINGO é $@wrap$. Ela recebe como argumentos os parâmetros *index* e *limit*, onde *limit* é o número de elementos do conjunto, que no exemplo citado pode ser determinado por $@size(turnos)$, e *index* é o índice considerado. $@wrap(index, limit) = index + k \times limit$, onde k é um inteiro tal que $index \in [1, limit]$. Informalmente, k deve ser tal que $@wrap(index, limit)$ devolva um número entre 1 e *limit*. Exemplo: $@wrap(8, 6) = 8 + (-1) \times 6 = 2$; $@wrap(0, 6) = 0 + (1) \times 6 = 6$; $@wrap(-1, 6) = -1 + (1) \times 6 = 5$, etc. Assim, no LINGO, as restrições $x_i + x_{i-1} \geq demanda_i \; \forall i \in turnos$ podem ser representadas por: $@for(turnos(i) : x(i) + x(@wrap(i-1, @size(turnos))) \geq demanda(i))$.

A seguir, a implementação LINGO do problema interfaceando com o Excel.

No arquivo Excel considerado há a seguinte correspondência de nomes para os blocos de células:

Bloco de células	Nome
B3:B8	turnos
E3:E8	custo
D3:D8	demanda
G3:G8	excturno
F3:F8	solução
E10	fo



	Turnos	Horários	Enfermeiros	Custo	Solução	ExcTurno
1						
2						
3	1	08 às 12	51	800	51	0
4	2	12 às 16	58	800	34	0
5	3	16 às 20	62	900	28	0
6	4	20 às 24	41	1000	13	0
7	5	24 às 04	32	1000	19	0
8	6	04 às 08	19	900	0	0
9						
10			Custo Total =	125200		

```
sets:
  turnos/@ole('enfermeiros.xls','turnos')/:c, !custo;
                                           x, !solução;
                                           demanda;!demanda;
endsets

data:
  c, demanda = @ole('enfermeiros.xls','custo','demanda');
enddata
```

```
[fo] min = @sum(turnos(i): c(i)*x(i));

@for(turnos(i):
 [excturno] x(i) + x(@wrap(i-1,@size(turnos))) >= demanda(i));

@for(turnos(i): @gin(x(i)));

data:
  @ole('enfermeiros.xls','solução','fo','excturno') = x, fo, excturno;
enddata
```

Como se observa, na solução ótima devem ser contratados: 51 enfermeiros para iniciarem o trabalho no turno 1, 34 no turno 2, 28 no turno 3, 13 no turno 4, 19 no turno 5 e nenhum no turno 6. O custo total mínimo com a contratação de enfermeiros é de R$125.200,00.

Observação:
Relativamente ao problema anterior, suponha que cada enfermeiro possa fazer hora-extra trabalhando mais 4 horas consecutivas além de sua jornada normal de trabalho, isto é, mais um turno de trabalho. Suponha que a hora-extra seja remunerada em 50% a mais que a hora normal. Considere, também, que em cada turno não mais de 20% dos enfermeiros possam estar fazendo hora-extra. Faça um modelo de programação linear inteira que minimize os gastos com a contratação de mão-de-obra. Antes de resolver o problema, pense na seguinte questão: A solução ótima dessa variante poderá ter custo menor que a da solução ótima sem a possibilidade de os enfermeiros fazerem hora-extra? Justifique.

Solução:

(a) Variáveis de decisão:
x_i = número de enfermeiros que iniciam sua jornada no início do turno i e não fazem hora-extra.
y_i = número de enfermeiros que iniciam sua jornada no início do turno i e fazem hora-extra.

(b) Função objetivo:

$$\begin{aligned} \min f(x,y) \quad = \quad & 800x_1 + 800x_2 + 900x_3 + 1000x_4 + 1000x_5 + 900x_6 + \\ & 1400y_1 + 1550y_2 + 1650y_3 + 1750y_4 + 1600y_5 + 1500y_6 \end{aligned}$$

(c) Restrições:

c.1) Número de enfermeiros necessários no turno 1:
$x_1 + x_6 + y_5 + y_1 + y_6 \geq 51$

c.2) Número de enfermeiros necessários no turno 2:
$x_2 + x_1 + y_6 + y_2 + y_1 \geq 58$

c.3) Número de enfermeiros necessários no turno 3:
$x_3 + x_2 + y_1 + y_3 + y_2 \geq 62$

c.4) Número de enfermeiros necessários no turno 4:
$x_4 + x_3 + y_2 + y_4 + y_3 \geq 41$

c.5) Número de enfermeiros necessários no turno 5:
$x_5 + x_4 + y_3 + y_5 + y_4 \geq 32$

c.6) Número de enfermeiros necessários no turno 6:
$x_6 + x_5 + y_4 + y_6 + y_5 \geq 19$

c.7) Limite de enfermeiros no turno 1:
$y_5 \leq 0,20(x_1 + x_6 + y_5 + y_1 + y_6)$

c.8) Limite de enfermeiros no turno 2:
$y_6 \leq 0,20(x_2 + x_1 + y_6 + y_2 + y_1)$

c.9) Limite de enfermeiros no turno 3:
$y_1 \leq 0,20(x_3 + x_2 + y_1 + y_3 + y_2)$

c.10) Limite de enfermeiros no turno 4:
$y_2 \leq 0,20(x_4 + x_3 + y_2 + y_4 + y_3)$

c.11) Limite de enfermeiros no turno 5:
$y_3 \leq 0,20(x_5 + x_4 + y_3 + y_5 + y_4)$

c.12) Limite de enfermeiros no turno 6:
$y_4 \leq 0,20(x_6 + x_5 + y_4 + y_6 + y_5)$

c.13) Integralidade e não-negatividade:
$x_1, x_2, x_3, x_4, x_5, x_6, y_1, y_2, y_3, y_4, y_5, y_6 \in \mathbb{Z}^+$

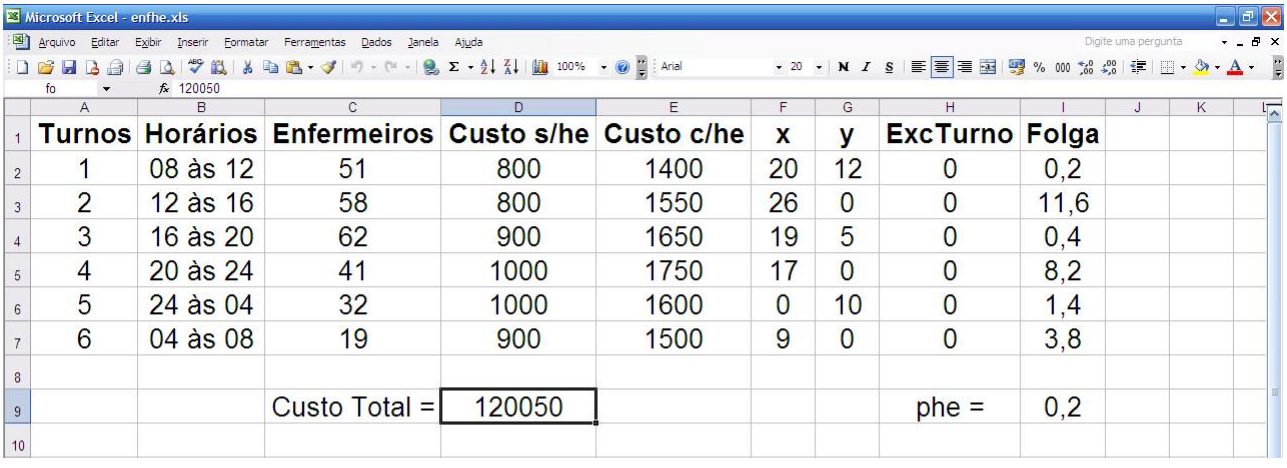
Para fazer um modelo genérico desse PPL, considere que *phe* significa o percentual máximo de enfermeiros fazendo hora-extra em cada turno, no caso, 0,20 e que c_i é o custo sem hora-extra e d_i, com hora-extra. Dessa forma, o modelo genérico para este problema pode ser formulado como:

$$\min \quad \sum_{i \in turnos} (c_i x_i + d_i y_i)$$

$$\begin{array}{rcll} x_i + x_{i-1} + y_{i-2} + y_i + y_{i-1} & \geq & demanda_i & \forall i \in turnos \\ y_{i-2} & \leq & phe \times (x_i + x_{i-1} + y_{i-2} + y_i + y_{i-1}) & \forall i \in turnos \\ x_i, y_i & \in & \mathbb{Z}^+ & \forall i \in turnos \end{array}$$

A seguir, o modelo LINGO interfaceando com um arquivo Excel referente ao problema em questão, com a seguinte correspondência de nomes para os blocos de células:

Bloco de células	Nome	Bloco de células	Nome
A2:A7	turnos	I9	*phe*
D2:D7	custoshe	I2:I7	folga
E2:E7	custoche	F2:F7	x
C2:C7	demanda	G2:G7	y
H2:H7	excturno	D9	fo



	A	B	C	D	E	F	G	H	I	J	K
1	**Turnos**	**Horários**	**Enfermeiros**	**Custo s/he**	**Custo c/he**	**x**	**y**	**ExcTurno**	**Folga**		
2	1	08 às 12	51	800	1400	20	12	0	0,2		
3	2	12 às 16	58	800	1550	26	0	0	11,6		
4	3	16 às 20	62	900	1650	19	5	0	0,4		
5	4	20 às 24	41	1000	1750	17	0	0	8,2		
6	5	24 às 04	32	1000	1600	0	10	0	1,4		
7	6	04 às 08	19	900	1500	9	0	0	3,8		
8											
9			Custo Total =	120050				phe =	0,2		
10											

```
sets:
  turnos/@ole('enfhe.xls','turnos')/:c, !custo sem hora-extra;
                                     d, !custo com hora-extra;
                                     x, !enfermeiros sem hora-extra;
                                     y, !enfermeiros com hora-extra;
                                     demanda;!demanda;
endsets

data:
  c, d, demanda = @ole('enfhe.xls','custoshe','custoche','demanda');
  phe = @ole('enfhe.xls','phe');
enddata

[fo] min = @sum(turnos(i): c(i)*x(i) + d(i)*y(i));

@for(turnos(i):
 [excturno] x(i) + x(@wrap(i-1,@size(turnos))) +
            y(i) + y(@wrap(i-1,@size(turnos))) +
            y(@wrap(i-2,@size(turnos))) >= demanda(i));


@for(turnos(i):
 [folga] y(@wrap(i-2,@size(turnos))) <= phe*
         (x(i) + x(@wrap(i-1,@size(turnos))) +
            y(i) + y(@wrap(i-1,@size(turnos))) +
            y(@wrap(i-2,@size(turnos)))));

@for(turnos(i):
      @gin(x(i));
      @gin(y(i)));

data:
  @ole('enfhe.xls','x','fo','excturno') = x, fo, excturno;
  @ole('enfhe.xls','y','folga') = y, folga;
enddata
```

Como se observa, na solução ótima devem ser contratados: 20 enfermeiros para iniciarem o trabalho no turno 1 sem fazer hora-extra, 26 no turno 2, 19 no turno 3, 17 no turno 4, nenhum no turno 5 e 9 no turno 6. Por outro lado, devem ser contratados 12 enfermeiros no turno 1 fazendo hora-extra, 5 no turno 3 e 10 no turno 5. O custo total mínimo com a contratação de enfermeiros é de R$120.050,00.

2.9 Problema da Fábrica de Prateleiras

Uma fábrica manufatura 5 tipos de prateleiras (p_1, p_2, p_3, p_4, p_5) utilizando dois processos de produção: processo normal (N) e processo acelerado (A). Cada produto requer um certo número de horas para ser trabalhado dentro de cada processo e alguns produtos só podem ser fabricados através de um dos tipos de processo. O quadro a seguir resume o consumo (em horas) dentro de cada esquema de fabricação e os lucros (em R$) obtidos após a dedução dos custos de produção.

Prateleiras	p_1	p_2	p_3	p_4	p_5
Lucro (R$/unidade)	570	575	555	550	560
Processo Normal	12	16	-	19	9
Processo Acelerado	10	16	5	-	-

A montagem final de cada prateleira requer 16 h de mão-de-obra por unidade. A fábrica possui 3 máquinas para o processo normal e 2 para o processo acelerado. As máquinas trabalham em 2 turnos de 8 horas por dia em um regime de 6 dias por semana. Uma equipe de 8 pessoas trabalha em turno único de 8 horas durante 6 dias na montagem das prateleiras. Determine o melhor esquema de produção.

Solução:

(a) Variáveis de decisão:
x_{ij} = número de prateleiras do tipo j a serem fabricadas pelo processo i, sendo i = N (processo normal) e i = A (processo acelerado).

(b) Função objetivo:
$\max f(x) = 570(x_{N1} + x_{A1}) + 575(x_{N2} + x_{A2}) + 555x_{A3} + 550x_{N4} + 560x_{N5}$

(c) Restrições:

c.1) Tempo disponível para o processo normal:
$12x_{N1} + 16x_{N2} + 19x_{N4} + 9x_{N5} \leq 288 \ \ (= 3 \times 2 \times 8 \times 6)$

c.2) Tempo disponível para o processo acelerado:
$10x_{A1} + 16x_{A2} + 5x_{A3} \leq 192 \ \ (= 2 \times 2 \times 8 \times 6)$

c.3) Tempo disponível para a montagem:
$16(x_{N1} + x_{A1} + x_{N2} + x_{A2} + x_{A3} + x_{N4} + x_{N5}) \leq 384 \ \ (= 8 \times 8 \times 6)$

c.4) Integralidade e não-negatividade:
$x_{N1}, x_{A1}, x_{N2}, x_{A2}, x_{A3}, x_{N4}, x_{N5} \in \mathbb{Z}^+$

Sejam os seguintes dados de entrada:

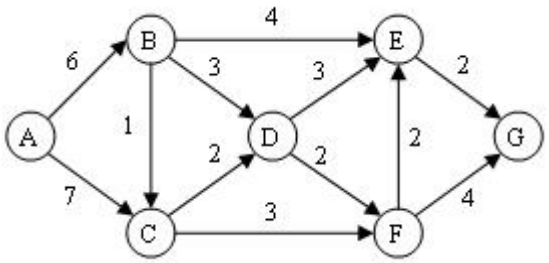
$Prat$	:	Conjunto das prateleiras
$Proc$	:	Conjunto dos tipos de processo
tp_i	:	Tempo disponível para a realização do processo i
t_{ij}	:	Tempo necessário para a produção de cada prateleira j no processo i
tm_j	:	Tempo necessário para a montagem de cada prateleira j
$Totaltm$	:	Tempo total disponível para a montagem das prateleiras

O modelo genérico pode ser formulado da seguinte forma:

$$
\begin{array}{lrcll}
\max & \displaystyle\sum_{i \in Proc} \sum_{j \in Prat} c_i x_{ij} & & & \\
& \displaystyle\sum_{j \in Prat} t_{ij} x_{ij} & \leq & tp_i & \forall i \in Proc \\
& \displaystyle\sum_{i \in Proc} \sum_{j \in Prat} tm_j x_{ij} & \leq & Totaltm & \\
& x_{ij} & \in & \mathbb{Z}^+ & \forall i \in Proc, \forall j \in Prat
\end{array}
$$

2.10 Fluxo Máximo em Redes

A figura a seguir representa uma rede de comunicação de dados entre computadores. Os números representam a capacidade máxima em MBytes por segundo que pode ser transmitido de um computador para outro. Admita que a transmissão só é possível no sentido especificado pela seta. Qual é o fluxo máximo que pode passar entre A e G através da rede?



Solução:

(a) Variáveis de decisão:
x_{ij} = quantidade de fluxo a ser enviada do nó i ao nó j, $j \neq i$.

(b) Função objetivo:
$\max f(x) = x_{AB} + x_{AC}$

(c) Restrições:

c.1) Equilíbrio de fluxo nos nós:

$$\begin{aligned}
x_{EG} + x_{FG} - (x_{AB} + x_{AC}) &= 0 \quad \text{(Nó A)} \\
x_{AB} - (x_{BC} + x_{BD} + x_{BE}) &= 0 \quad \text{(Nó B)} \\
x_{AC} + x_{BC} - (x_{CD} + x_{CF}) &= 0 \quad \text{(Nó C)} \\
x_{BD} + x_{CD} - (x_{DE} + x_{DF}) &= 0 \quad \text{(Nó D)} \\
x_{BE} + x_{DE} + x_{FE} - x_{EG} &= 0 \quad \text{(Nó E)} \\
x_{CF} + x_{DF} - (x_{FE} + x_{FG}) &= 0 \quad \text{(Nó F)}
\end{aligned}$$

c.2) Capacidade nos arcos:

$$\begin{aligned}
x_{AB} &\leq 6 \\
x_{AC} &\leq 7 \\
x_{BC} &\leq 1 \\
x_{BD} &\leq 3 \\
x_{BE} &\leq 4 \\
x_{CD} &\leq 2 \\
x_{CF} &\leq 3 \\
x_{DE} &\leq 3 \\
x_{DF} &\leq 2 \\
x_{EG} &\leq 2 \\
x_{FE} &\leq 2 \\
x_{FG} &\leq 4
\end{aligned}$$

c.3) Integralidade e não-negatividade:
$x_{AB}, x_{AC}, x_{BC}, x_{BD}, x_{BE}, x_{CD}, x_{CF}, x_{DE}, x_{DF}, x_{EG}, x_{FE}, x_{FG} \in \mathbb{Z}^+$

Para formular o problema de forma genérica, considere as seguintes notações:

V : Conjunto de nós
cap_{ij} : Capacidade do arco (i, j)
n : Cardinalidade do conjunto de nós, isto é, $n = |V|$

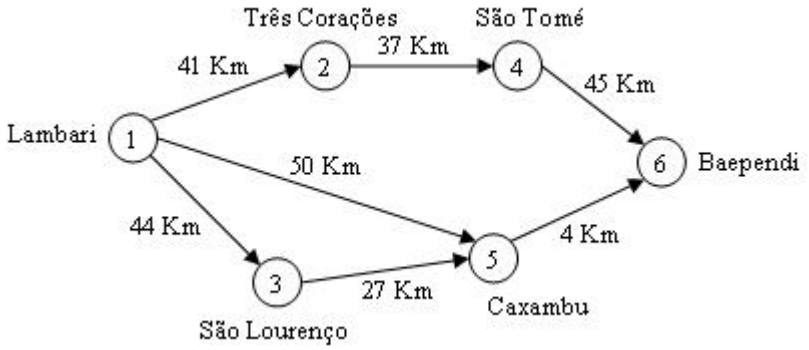
Assim, o modelo genérico pode ser escrito como:

$$\begin{array}{lrcll} \max & \sum\limits_{j \in V} x_{1j} & & & \\ & \sum\limits_{j \in V} x_{ij} - \sum\limits_{j \in V} x_{ji} & = & 0 & \forall i \in V, i \neq 1 \text{ e } i \neq n \\ & \sum\limits_{j \in V} x_{1j} - \sum\limits_{i \in V} x_{in} & = & 0 & \\ & x_{ij} & \leq & cap_{ij} & \forall i, j \in V \\ & x_{ij} & \in & \mathbb{Z}^+ & \forall i, j \in V \end{array}$$

Neste modelo, considera-se que o nó 1 é o nó origem e que o nó n é o nó destino.

2.11 Caminho Mínimo

[Extraído de Lachtermacher (2004), vide [4]] Uma fábrica de artigos de decoração, localizada em Lambari (MG), deve entregar uma grande quantidade de peças na cidade de Baependi (MG). A empresa quer saber qual o caminho que seu caminhão de entregas deve fazer para minimizar a distância total percorrida. A figura a seguir, extraída de Lachtermacher (2004), representa, na forma de rede, as ligações entre as cidades da região.



Solução:

(a) Variáveis de decisão:
$x_{ij} = \begin{cases} 1, & \text{se o arco } (i, j) \text{ pertencer ao caminho;} \\ 0, & \text{caso contrário.} \end{cases}$

(b) Função objetivo:
$\min f(x) = 41x_{12} + 44x_{13} + 50x_{15} + 37x_{24} + 27x_{35} + 45x_{46} + 4x_{56}$

(c) Restrições:

c.1) Equilíbrio de fluxo nos nós:

$$\begin{array}{lll} x_{12} + x_{13} + x_{15} & = 1 & \text{(Nó 1)} \\ x_{12} - x_{24} & = 0 & \text{(Nó 2)} \\ x_{13} - x_{35} & = 0 & \text{(Nó 3)} \\ x_{24} - x_{46} & = 0 & \text{(Nó 4)} \\ x_{15} + x_{35} - x_{56} & = 0 & \text{(Nó 5)} \\ x_{46} + x_{56} & = 1 & \text{(Nó 6)} \end{array}$$

c.2) Integralidade e não-negatividade:
$x_{12}, x_{13}, x_{15}, x_{24}, x_{35}, x_{46}, x_{56} \in \{0, 1\}$

Considere os seguintes parâmetros de entrada:

V : Conjunto dos vértices (nós)
d_{ij} : Distância do vértice i ao vértice j
n : Cardinalidade do conjunto de vértices, isto é, $n = |V|$

Considerando o nó 1 como o nó origem e n como o nó destino, pode-se modelar o problema de caminho mínimo genericamente como:

$$
\begin{array}{lrcll}
\min & \displaystyle\sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} & & & \\
& \displaystyle\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} & = & 0 & \forall i \in V, i \neq 1 \text{ e } i \neq n \\
& \displaystyle\sum_{j \in V} x_{1j} & = & 1 & \\
& \displaystyle\sum_{i \in V} x_{in} & = & 1 & \\
& x_{ij} & \in & \{0, 1\} & \forall i, j \in V
\end{array}
$$

2.12 Programação da produção - exemplo 1

Uma determinada empresa está interessada em maximizar o lucro mensal proveniente de quatro de seus produtos, designados por I, II, III e IV. Para fabricar esses produtos, ela utiliza dois tipos de máquinas (M1 e M2) e dois tipos de mão-de-obra (MO1 e MO2), que têm as seguintes disponibilidades:

Máquina	Disponibilidade (máquina-hora/mês)
M1	70
M2	20

Mão-de-obra	Disponibilidade (homem-hora/mês)
MO1	120
MO2	160

O setor técnico da empresa fornece os seguintes coeficientes, que especificam o total de horas de máquina e horas de mão-de-obra necessárias para a produção de uma unidade de cada produto:

	Produtos			
Máquinas	I	II	III	IV
M1	5	4	8	9
M2	2	6	0	8

	Produtos			
Mão-de-obra	I	II	III	IV
MO1	2	4	2	8
MO2	7	3	0	7

O setor comercial fornece as seguintes informações:

Produtos	Potencial de venda (unidades/mês)	Lucro unitário (R$/mês)
I	70	10
II	60	8
III	40	9
IV	20	7

Faça o planejamento da produção mensal da empresa objetivando maximizar o lucro.

Solução:

(a) Variáveis de decisão:
x_j = quantidade de produtos do tipo j a serem fabricadas mensalmente.

(b) Função objetivo:
$\max f(x) = 10x_I + 8x_{II} + 9x_{III} + 7x_{IV}$

(c) Restrições:

c.1) Disponibilidade de tempo da máquina 1:
$5x_I + 4x_{II} + 8x_{III} + 9x_{IV} \leq 70$

c.2) Disponibilidade de tempo da máquina 2:
$2x_I + 6x_{II} + 8x_{IV} \leq 20$

c.3) Disponibilidade de tempo da mão-de-obra 1:
$2x_I + 4x_{II} + 2x_{III} + 8x_{IV} \leq 120$

c.4) Disponibilidade de tempo da mão-de-obra 2:
$7x_I + 3x_{II} + 7x_{IV} \leq 160$

c.5) Potencial de venda dos produtos:
$x_I \leq 70$
$x_{II} \leq 60$
$x_{III} \leq 40$
$x_{IV} \leq 20$

c.6) Integralidade e não-negatividade:
$x_I, x_{II}, x_{III}, x_{IV} \in \mathbb{Z}^+$

Para a modelagem genérica do problema, considere os seguintes dados de entrada, além da variável de decisão apresentada anteriormente:

$Prod$	:	Conjunto dos produtos
Maq	:	Conjunto das máquinas
MO	:	Conjunto das mãos-de-obra
l_j	:	Lucro de uma unidade do produto j, em unidades monetárias
p_j	:	Potencial de venda do produto j
m_{ij}	:	Tempo gasto para a produção do produto j na máquina i, em horas
o_{kj}	:	Tempo requerido para a produção do produto j pela mão-de-obra k, em horas
tm_i	:	Tempo disponível da máquina i, em horas
to_k	:	Tempo disponível para a mão-de-obra k, em horas

Então, o modelo de programação matemática que maximiza a produção é:

$$
\begin{aligned}
\max \quad & \sum_{j \in Prod} l_j x_j & & & \\
& \sum_{j \in Prod} m_{ij} x_j & \leq & tm_i & \forall i \in Maq \\
& \sum_{j \in Prod} o_{kj} x_j & \leq & to_k & \forall k \in MO \\
& x_j & \leq & p_j & \forall j \in Prod \\
& x_j & \in & \mathbb{Z}^+ & \forall j \in Prod
\end{aligned}
$$

2.13 Sequenciamento em processadores paralelos e idênticos

Suponha que seja necessário executar uma lista de 10 *jobs* em um conjunto de 3 processadores. Sabe-se que cada *job* pode ser executado em qualquer ordem e em qualquer processador, sendo o tempo de processamento independente do processador. O tempo (em minutos) gasto para execução de cada *job* é: 6, 4, 5, 4, 3, 7, 8, 5, 3 e 3. Elabore o modelo de programação matemática que minimize o tempo de execução de todos os *jobs*.

Solução:

(a) Variáveis de decisão:

$x_{ij} = \begin{cases} 1, & \text{se o } job\ j \text{ for executado no processador } i; \\ 0, & \text{caso contrário.} \end{cases}$

C_{max} = *makespan* (Instante de término do processador mais carregado)

(b) Função objetivo:

$\min C_{max}$

(c) Restrições:

c.1) Cada *job* deve ser executado em um único processador:

$x_{11} + x_{21} + x_{31} = 1 \quad (job\ 1)$
$x_{12} + x_{22} + x_{32} = 1 \quad (job\ 2)$
$x_{13} + x_{23} + x_{33} = 1 \quad (job\ 3)$
$x_{14} + x_{24} + x_{34} = 1 \quad (job\ 4)$
$x_{15} + x_{25} + x_{35} = 1 \quad (job\ 5)$
$x_{16} + x_{26} + x_{36} = 1 \quad (job\ 6)$
$x_{17} + x_{27} + x_{37} = 1 \quad (job\ 7)$
$x_{18} + x_{28} + x_{38} = 1 \quad (job\ 8)$
$x_{19} + x_{29} + x_{39} = 1 \quad (job\ 9)$
$x_{1,10} + x_{2,10} + x_{3,10} = 1 \quad (job\ 10)$

c.2) Tempo de execução dos *jobs* em cada processador:

Proc. 1: $6x_{11} + 4x_{12} + 5x_{13} + 4x_{14} + 3x_{15} + 7x_{16} + 8x_{17} + 5x_{18} + 3x_{19} + 3x_{1,10} \leq C_{max}$
Proc. 2: $6x_{21} + 4x_{22} + 5x_{23} + 4x_{24} + 3x_{25} + 7x_{26} + 8x_{27} + 5x_{28} + 3x_{29} + 3x_{2,10} \leq C_{max}$
Proc. 3: $6x_{31} + 4x_{32} + 5x_{33} + 4x_{34} + 3x_{35} + 7x_{36} + 8x_{37} + 5x_{38} + 3x_{39} + 3x_{3,10} \leq C_{max}$

c.3) Integralidade e não-negatividade:

$x_{ij} \in \{0, 1\}\ \forall i = 1, 2, 3$ e $\forall j = 1, 2, ..., 10$

Considere as seguintes notações:

$Jobs$: Conjunto dos *jobs*
$Procs$: Conjunto dos processadores
t_j : Tempo de execução do *job* j

Genericamente, pode-se modelar o problema como:

$$\begin{array}{lrcll} \min & C_{max} & & & \\ & \sum\limits_{i \in Procs} x_{ij} & = & 1 & \forall j \in Jobs \\ & \sum\limits_{j \in Jobs} t_j x_{ij} & \leq & C_{max} & \forall i \in Procs \\ & x_{ij} & \in & \{0, 1\} & \forall i \in Procs \text{ e } \forall j \in Jobs \end{array}$$

2.14 Planejamento da Produção - Problema da Fábrica de Motores

[Retirado de Lachtermacher (2004), vide [4]] A LCL Motores recebeu recentemente uma encomenda para produzir três tipos de motores. Cada tipo de motor necessita de um determinado número de horas de trabalho no setor de montagem e acabamento. A LCL pode terceirizar parte de sua produção. A tabela a seguir resume essas informações:

Modelo	1	2	3	Total (h)
Demanda (unidades)	3000	2500	500	
Montagem (h/unidade)	1,1	1,9	0,7	6000
Acabamento (h/unidade)	2,5	0,8	4,0	10000
Custo de produção (R$)	50	90	120	
Terceirizado (R$)	65	92	140	

Elabore o modelo de programação matemática que minimiza os custos de produção.

Solução:

(a) Variáveis de decisão:
x_i = quantidade de motores do modelo i a ser produzido.
y_i = quantidade de motores do modelo i a ser terceirizado.

(b) Função objetivo:
$\min f(x, y) = 50x_1 + 90x_2 + 120x_3 + 65y_1 + 92y_2 + 140y_3$

(c) Restrições:

c.1) Atendimento à demanda dos modelos de motores:
$x_1 + y_1 \geq 3000$
$x_2 + y_2 \geq 2500$
$x_3 + y_3 \geq 500$

c.2) Respeito ao tempo disponível para a montagem:
$1,1x_1 + 1,9x_2 + 0,7x_3 \leq 6000$

c.3) Respeito ao tempo disponível para o acabamento:
$2,5x_1 + 0,8x_2 + 4x_3 \leq 10000$

c.4) Integralidade e não-negatividade:
$x_1, x_2, x_3, y_1, y_2, y_3 \in \mathbb{Z}^+$

Considere que:

$Modelos$: Conjunto dos diferentes modelos de motores
$cprod_j$: Custo de produção do modelo j de motor
$cterc_j$: Custo de terceirização do modelo j de motor
$demanda_j$: Demanda dos motores do modelo j
$TempMont_j$: Tempo necessário para a montagem do modelo j de motor
$TempAcab_j$: Tempo necessário para o acabamento do modelo j de motor
$TempDispMont$: Tempo disponível para a montagem
$TempDispAcab$: Tempo disponível para o acabamento

Genericamente, pode-se modelar o problema da seguinte forma:

$$
\begin{aligned}
\min \quad & \sum_{j \in Modelos} cprod_j x_j + \sum_{j \in Modelos} cterc_j y_j & & & \\
& x_j + y_j & \geq & demanda_j & \forall j \in Modelos \\
& \sum_{j \in Modelos} TempMont_j x_j & \leq & TempDispMont & \\
& \sum_{j \in Modelos} TempAcab_j x_j & \leq & TempDispAcab & \\
& x_j,\, y_j & \in & \mathbb{Z}^+ & \forall j \in Modelos
\end{aligned}
$$

Segue um modelo LINGO que lê os dados do arquivo-texto Motores.txt e exporta a solução para os arquivos Producao.txt, no caso dos modelos produzidos pela própria empresa, e Terceirizado.txt, no caso dos modelos que devem ser terceirizados.

```
sets:
  Modelos / @file('Motores.txt') /: x, y, cprod, cterc,
                                    TempMont, TempAcab,
                                    demanda;
endsets

! Importar dados de arquivo texto;
data:
  demanda = @file('Motores.txt');
  TempMont = @file('Motores.txt');
  TempAcab = @file('Motores.txt');
  cprod = @file('Motores.txt');
  cterc = @file('Motores.txt');
  TempDispMont = @file('Motores.txt');
  TempDispAcab = @file('Motores.txt');
enddata

[fo] min = @sum(Modelos(j): cprod(j)*x(j)) +
           @sum(Modelos(j): cterc(j)*y(j));


[folgaMont] @sum(Modelos(j): TempMont(j)*x(j)) <= TempDispMont;

[folgaAcab] @sum(Modelos(j): TempAcab(j)*x(j)) <= TempDispAcab;

@for(Modelos(j):
  [Estoque]  x(j) + y(j) >= demanda(j));

@for(Modelos(j):
     @gin(x(j));
     @gin(y(j)));

! Exportar para arquivo texto;
data:
  @text('Producao.txt') = x;
  @text('Terceirizado.txt') = y;
enddata
```

Neste modelo, 'Estoque' é um vetor de 3 posições e indica o excesso de produção; 'folgaMont' é um escalar que indica a folga, em horas, no setor de montagem e 'folgaAcab' é um escalar que indica a folga, em horas, no setor de acabamento.

O arquivo Motores.txt, que contém os dados, é apresentado a seguir. Observe que a cada chamada desse arquivo são lidos todos os dados até o símbolo *til* (∼). Assim, a ordem de chamada deve ser aquela referente à leitura dos dados no modelo. Observe também que, no caso de números decimais, deve-se utilizar o ponto como separador decimal e não a vírgula.

```
! Modelos;
1 2 3 ~

! Demanda;
3000 2500 500 ~

! Montagem;
1.1 1.9 0.7 ~

! Acabamento;
2.5 0.8 4 ~

! Custo de produção;
50 90 120 ~

! Custo de terceirização;
65 92 140 ~

! Tempo disponivel para montagem;
6000 ~

! Tempo disponível para acabamento;
10000
```

O custo ótimo de produção neste exemplo é de $438.750,00 unidades monetárias, sendo que na solução ótima devem ser produzidas pela própria fábrica 3000 unidades do modelo 1, 625 do modelo 2 e 500 do modelo 3; enquanto que 1875 unidades do modelo 2 devem ser terceirizadas. Observe que o gargalo do sistema produtivo é o setor de acabamento, já que na solução ótima não há folga nesse setor; já na montagem há 1162,5 horas de folga.

2.15 Problema de empacotamento (*Bin Packing*)

Há um conjunto de contêineres e outro de itens a serem alocados a esses contêineres. Sabe-se que cada contêiner i tem capacidade cap_i e que cada item j tem um peso w_j. Determine o número mínimo de contêineres necessário para empacotar todos os itens.

Solução:

Para a formulação de programação matemática deste problema, sejam os seguintes parâmetros de entrada:

Conteineres:		Conjunto dos contêineres
Itens	:	Conjunto dos itens
w_j	:	Peso do item j
cap_i	:	Capacidade do contêiner i

e as seguintes variáveis de decisão:

y_i	:	Variável que assume o valor 1 se o contêiner i é usado e 0, caso contrário
x_{ij}	:	Variável que assume o valor 1 se o item j é colocado no contêiner i e 0, caso contrário

Então o *Bin Packing Problem* pode ser modelado como:

$$\min \sum_{i \in Conteineres} y_i \qquad \text{(BP1)}$$

$$\sum_{i \in Conteineres} x_{ij} = 1 \qquad \forall j \in Itens \qquad \text{(BP2)}$$

$$\sum_{j \in Itens} w_j x_{ij} \leq cap_i y_i \qquad \forall i \in Conteineres \qquad \text{(BP3)}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in Conteineres,\ \forall j \in Itens \qquad \text{(BP4)}$$

$$y_i \in \{0,1\} \qquad \forall i \in Conteineres \qquad \text{(BP5)}$$

Neste modelo, a expressão (BP1) indica que o objetivo é minimizar o número de contêineres. As restrições (BP2) asseguram que cada item é alocado a um único contêiner. O conjunto de restrições (BP3) impede que a capacidade de cada contêiner seja ultrapassada. As restrições (BP4) e (BP5) estabelecem que as variáveis de decisão são binárias.

2.16 *Open Dimensional Problem*

Considere um objeto retangular de largura W e comprimento suficientemente grande. Considere, também, a necessidade de se obter, a partir desse objeto, um conjunto de itens diversos $i = 1, \cdots, n$ de dimensões (h_i, w_i), onde h_i é o comprimento e $w_i \leq W$, a largura (Vide Figura 1).

O *Open Dimensional Problem* (ODP) consiste em determinar a melhor forma de cortar o objeto para atender ao pedido, de modo a minimizar o comprimento do objeto utilizado.

A Figura 2 mostra uma solução para o exemplo da Figura 1, a qual, naturalmente, não é a melhor. Nesta solução, o comprimento do objeto é dado pela soma das alturas dos itens 5, 1 e 6, isto é, o comprimento utilizado é $h_5 + h_1 + h_6$.

Este problema aparece na literatura com uma série de nomes diferentes, dependendo do tipo de variação sofrida. No caso de os itens serem retângulos, o ODP recebe os nomes de *Rectangular Strip Packing Problem* ou *Two-Dimensional Strip Packing Problem*. Quando os itens retangulares devem ser colocados no objeto de forma ortogonal, o ODP é conhecido como *Orthogonal*

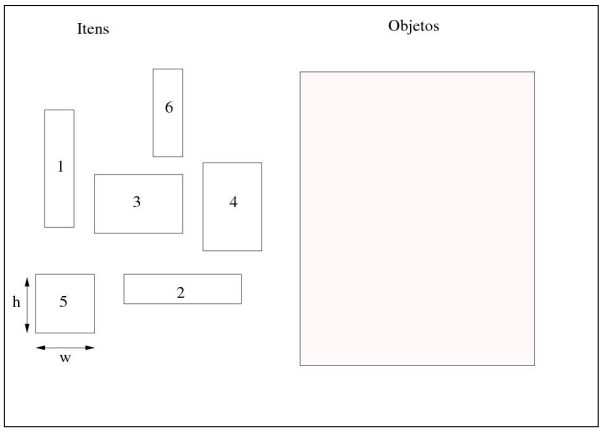


Figura 1: Objeto e itens a serem produzidos

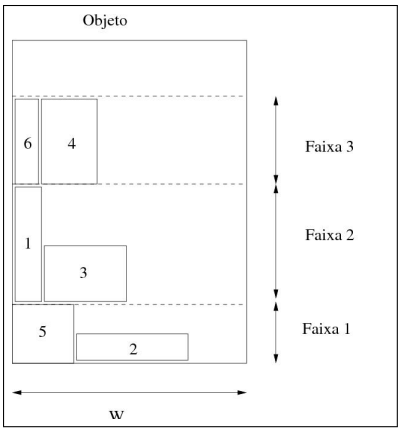


Figura 2: Exemplo de uma solução

Rectangular Strip Packing Problem. Ele é denominado *Level Packing Problem* quando os itens são alocados no objeto formando-se níveis. Quando os itens possuem forma não-regular, como, por exemplo, na indústria de sapatos, o ODP é referenciado como *Irregular Strip Packing Problem* ou *Nesting Problem*. No caso de itens retangulares, existem duas formas de se cortar os objetos: de forma guilhotinada e de forma não-guilhotinada. Na forma guilhotinada, o corte se estende de um lado ao outro do objeto. Já na forma não guilhotinada, o corte acompanha o contorno dos itens. O modelo apresentado a seguir considera apenas a forma guilhotinada.

Solução:

No modelo a seguir, os itens são alocados formando-se faixas e o objetivo consiste em minimizar a soma dos comprimentos das faixas. Para sua modelagem, considera-se que:

- o primeiro item alocado em cada faixa (mais à esquerda) é o dentre os outros
- a primeira faixa do objeto (mais embaixo) é a de maior altura
- os itens são ordenados de forma decrescente em relação à altura, ou seja, $h_1 \geq h_2 \geq \cdots \geq h_n$

Como consequencia da terceira consideração, tem-se que a altura de cada faixa corresponde à altura h_i do item i que a inicializa (primeiro item alocado).

Assim, sendo n o número de faixas formadas para alocar todos os itens demandados, pode-se definir a seguinte variável de decisão:

$$y_i = \begin{cases} 1 & \text{Se o item } i \text{ inicializa a faixa } i \\ 0 & \text{Caso contrário} \end{cases}$$

Deve-se ressaltar, ainda, que, devido à primeira e terceira consideração, somente os itens j, tal que $j > i$, podem ser alocados na faixa. Essa condição se deve ao fato de que, se um item j, tal que $j = i$, inicializa a faixa i, ele não pode ser atribuído novamente a essa faixa. Assim sendo, é definida a seguinte variável binária:

$$x_{ij} = \begin{cases} 1 & \text{Se o item } j \text{ estiver alocado na faixa } i \\ 0 & \text{Caso contrário} \end{cases}$$

Assim, o *ODP* na forma guilhotinada pode ser modelado por:

$$
\begin{array}{lrcll}
\min & \sum_{i=1}^{n} h_i y_i & & & \\
 & \sum_{i=1}^{j-1} x_{ij} + y_j & = & 1 & \forall j = 1, \cdots, n \\
 & \sum_{j=i+1}^{n} w_j x_{ij} & \leq & (W - w_i)\, y_i & \forall i = 1, \cdots, n-1 \\
 & y_i & \in & \{0,1\} & \forall i = 1, \cdots, n \\
 & x_{ij} & \in & \{0,1\} & \forall i = 1, \cdots, n, \quad \forall j = 1, \cdots, n, \quad j < i
\end{array}
$$

O primeiro conjunto de restrições garante que cada item será alocado uma única vez. O segundo conjunto de restrições assegura que o somatório da largura dos itens alocados em cada faixa não ultrapassará a largura do objeto. As demais restrições definem as variáveis de decisão como binárias.

2.17 Programação de horários em escolas (*School timetabling*)

Há um conjunto P de professores, um conjunto T de turmas, um conjunto D de dias da semana (segunda, terça, quarta, quinta, sexta) e um conjunto H de horários diários para a realização de aulas. Suponha que a cada professor i está associada uma determinada matéria previamente alocada e que um professor i não pode dar mais do que $q = 2$ aulas por dia para qualquer turma. Suponha, também, que a cada professor i está associada uma preferência c_{ikl} por dar aula em um determinado dia k e horário l, onde $c_{ikl} \in \{0, 1, 2, \cdots, 10\}$ e quanto maior o valor de c_{ikl} maior a preferência. Seja CH_{ij} a carga horária do professor i para a turma j. Elabore um modelo de programação inteira que faça a alocação dos professores às turmas maximizando a preferência dos professores com relação aos dias e horários das aulas.

Solução:

O problema de programação de horários em escolas pode ser modelado da seguinte forma:

$$
\begin{array}{lrcll}
\max & \sum_{i\in P}\sum_{j\in T}\sum_{k\in D}\sum_{l\in H} c_{ikl} x_{ijkl} & & & & \text{(PH1)} \\
 & \sum_{j\in T} x_{ijkl} & \leq & 1 & \forall i \in P,\ \forall k \in D,\ \forall l \in H & \text{(PH2)} \\
 & \sum_{i\in P} x_{ijkl} & \leq & 1 & \forall j \in T,\ \forall k \in D,\ \forall l \in H & \text{(PH3)} \\
 & \sum_{k\in D}\sum_{l\in H} x_{ijkl} & = & CH_{ij} & \forall i \in P,\ \forall j \in T & \text{(PH4)} \\
 & \sum_{l\in H} x_{ijkl} & \leq & q & \forall i \in P,\ \forall j \in T,\ \forall k \in D & \text{(PH5)} \\
 & x_{ijkl} & \in & \{0,1\} & \forall i \in P,\ \forall j \in T,\ \forall k \in D,\ \forall l \in H & \text{(PH6)}
\end{array}
$$

Neste modelo, as restrições (PH2) asseguram que fixado um dia e um horário diário, um professor não dá aula para mais de uma turma ao mesmo tempo. As restrições (PH3) impedem que uma turma tenha aula com mais de um professor ao mesmo tempo. As restrições (PH4) garantem que a carga horária de cada professor para cada turma é cumprida. O conjunto de restrições (PH5) impede que uma turma tenha mais de q aulas diárias com um mesmo professor, enquanto as restrições (PH6) estabelecem que as variáveis de decisão são binárias. Finalmente, (PH1) indica que a preferência dos professores é maximizada.

Observa-se que neste modelo as restrições (PH5) não indicam que as q aulas são consecutivas. Para modelar este caso, vide trabalho [7], disponível no endereço eletrônico:

http://www.decom.ufop.br/prof/marcone/Publicacoes/tesemarcone.ps

Uma restrição comum neste tipo de problema é considerar a indisponibilidade do professor. Neste caso, seja $disp_{ikl} = 1$ se o professor i está disponível no dia k e horário diário l e 0, caso contrário. Assim, as restrições (PH2) devem ser substituídas por:

$$\sum_{j \in T} x_{ijkl} \quad \leq \quad disp_{ikl} \quad \forall i \in P,\ \forall k \in D,\ \forall l \in H \quad \text{(PH7)}$$

2.18 Localização

[Retirado de Arenales *et al.* (2007). Vide [1]] A localização de facilidades é um aspecto crítico do planejamento estratégico de empresas privadas e públicas. Exemplos típicos no setor público envolvem decisões de localização de centros de saúde, escolas e estações de bombeiros, enquanto no setor privado tem-se a localização de fábricas, armazéns e centros de distribuição. Em diversas situações, tais como em sistemas de distribuição, as decisões da localização de facilidades e de designação de clientes a facilidades são feitas simultaneamente.

A seguir, apresentam-se modelos matemáticos de alguns problemas importantes de localização. Para tal, considere os seguintes parâmetros:

J : Conjunto de nós j que representam os clientes
I : Conjunto de locais i candidatos à localização de facilidades
q_j : Demanda do cliente j
d_{ij} : Distância do cliente j à facilidade localizada em i
c_{ij} : Custo de atender a demanda q_j do cliente j a partir de uma facilidade localizada em i
f_i : Custo fixo de instalação de uma facilidade no local i
Q_i : Capacidade da facilidade instalada no local i

2.18.1 p-Medianas

Este problema envolve a localização de p facilidades e a designação de clientes a facilidades de modo a minimizar a soma das distâncias de clientes a facilidades e tal que cada cliente seja atendido por uma única facilidade.
Variáveis de decisão:

$$x_{ij} = \begin{cases} 1, & \text{se o cliente } j \text{ é atendido pela facilidade localizada em } i; \\ 0, & \text{caso contrário.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{se a facilidade é aberta no local } i; \\ 0, & \text{caso contrário.} \end{cases}$$

O modelo genérico relativo ao problema em questão pode ser assim formulado:

$$\begin{aligned}
\min \quad & \sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij} & & & & \text{(LF1)} \\
& \sum_{i \in I} x_{ij} & = & 1 & \forall j \in J & \text{(LF2)} \\
& x_{ij} & \leq & y_i & \forall i \in I,\ \forall j \in J & \text{(LF3)} \\
& \sum_{i \in I} y_i & = & p & & \text{(LF4)} \\
& x_{ij} & \in & \{0,1\} & \forall i \in I,\ \forall j \in J & \text{(LF5)} \\
& y_i & \in & \{0,1\} & \forall i \in I & \text{(LF6)}
\end{aligned}$$

A função objetivo (LF1) minimiza a distância total de designação de clientes a facilidades. As restrições (LF2) garantem que cada cliente j é atendido por uma única facilidade. As restrições (LF3) asseguram que cada cliente j só pode ser designado a uma facilidade que esteja aberta no local i. A restrição (LF4) indica que exatamente p facilidades são abertas. As restrições (LF5) e (LF6) representam o tipo das variáveis.

É importante observar que as restrições (LF3) fazem a ligação entre as variáveis x e y. Sem elas, poderíamos ter, olhando apenas para (LF4), uma solução do tipo $y_2 = 1$ e $y_4 = 1$, indicando que seriam instaladas facilidades nos locais 2 e 4. Poderíamos ter, também, por (LF2), que $x_{13} = 1$ e $x_{11} = 1$, isto é, que os clientes 1 e 3 seriam atendidos por uma facilidade instalada no local 1, o que não é correto, pois nenhuma facilidade está instalada (aberta) no local 1. Este exemplo mostra a necessidade das restrições (LF4). Por elas, dado um cliente j e uma facilidade i, se $y_i = 1$ então x_{ij} pode ser 0 ou 1, isto é, o cliente j pode ou não ser atendido pela facilidade i. No entanto, se $y_i = 0$ então x_{ij} só pode ser 0, isto é, se não houver uma facilidade instalada (aberta) no local i, então nenhum cliente j pode ser atendido por i.

2.18.2 p-Centros

[Retirado de Arenales *et al.* (2007). Vide [1]] Este problema envolve a localização de p facilidades e a designação de clientes a facilidades de modo a minimizar a distância máxima de clientes a facilidades. Este problema admite variações do modelo básico. O problema de p-centros-nós restringe os nós de facilidades aos nós de clientes, enquanto o problema de p-centros-absolutos permite que os nós de facilidades estejam em qualquer lugar dos arcos que ligam nós de clientes.

Para formular este problema, considere as variáveis do problema das p-medianas e a seguinte variável adicional:

r : Distância máxima de um cliente quando designado a uma facilidade

O modelo de programação inteira referente a esse problema pode ser formulado como:

$$
\begin{array}{lrcll}
\min & r & & & & \text{(LF6)} \\
& r & \geq & \sum_{i \in I} d_{ij} x_{ij} & \forall j \in J & \text{(LF7)} \\
& \sum_{i \in I} x_{ij} & = & 1 & \forall j \in J & \text{(LF2)} \\
& x_{ij} & \leq & y_i & \forall i \in I,\ \forall j \in J & \text{(LF3)} \\
& \sum_{i \in I} y_i & = & p & & \text{(LF4)} \\
& x_{ij} & \in & \{0,1\} & \forall i \in I,\ \forall j \in J & \text{(LF5)} \\
& y_i & \in & \{0,1\} & \forall i \in I & \text{(LF6)}
\end{array}
$$

A função objetivo (LF6) minimiza a distância máxima de um cliente a uma facilidade. A restrição (LF7) expressa r como um limitante superior da distância de cada cliente j a uma facilidade. As demais restrições são idênticas às do problema das p-medianas.

2.18.3 p-Medianas capacitado

Neste caso, associa-se uma capacidade Q_i à facilidade instalada no local i. Assim, as restrições (LF3) são substituídas por:

$$\sum_{j \in J} q_j x_{ij} \quad \leq \quad Q_i y_i \quad \forall i \in I \quad \text{(LF8)}$$

2.19 Mistura de Minérios com Metas de Qualidade

Uma mineradora recebe uma encomenda para produzir 6000 toneladas de minério atendendo a especificação abaixo.

Elemento químico	Teor Mínimo permitido	Meta	Teor Máximo permitido
Fe (%)	44,5	47,0	49,5
Al_2O_3 (%)	0,27	0,32	0,37
P (%)	0,035	0,040	0,043
PPC (%)	2,05	2,35	2,65
He (%)	38	40	50

Sabe-se que esta encomenda pode ser atendida a partir de um conjunto de pilhas de minérios, cuja composição e disponibilidade são relacionadas a seguir.

Pilha	Fe (%)	Al_2O_3 (%)	P (%)	PPC (%)	He (%)	Massa (ton)
01	52,64	0,52	0,084	4,48	45	1500
02	39,92	0,18	0,029	0,65	97	2000
03	47,19	0,50	0,050	2,52	52	1700
04	49,36	0,22	0,039	1,74	78	1450
05	43,94	0,46	0,032	2,36	41	1250
06	48,97	0,54	0,057	4,34	90	1890
07	47,46	0,20	0,047	5,07	9	1640
08	46,52	0,32	0,039	3,51	4	1124
09	56,09	0,95	0,059	4,10	80	1990
10	46,00	0,26	0,031	2,51	21	900
11	49,09	0,22	0,040	4,20	12	1540
12	49,77	0,20	0,047	4,81	12	1630
13	53,03	0,24	0,047	4,17	1	1320
14	52,96	0,29	0,052	4,81	1	1245
15	42,09	0,17	0,031	1,38	47	1859

A tabela a seguir classifica os parâmetros de controle em 5 critérios: Irrelevante (-), Importante (I), Muito Importante (MI), Crítico (C) e Muito Crítico (MC), cujos pesos são também apresentados.

Critério	**-**	**I**	**MI**	**C**	**MC**
Peso do Critério	0	1	5	10	100
Parâmetro	Fe	Al_2O_3	P	PPC	He
Critério	MI	-	MC	C	-

Considere, ainda, os seguintes pesos para comparar os diversos parâmetros de controle entre si:

Parâmetro	Fe	Al_2O_3	P	PPC	He
Peso de comparação	1	100	1000	10	1

Suponha que se possa retomar apenas múltiplos de 10 toneladas e que para cada pilha só se pode retomar um mínimo de 500 toneladas. Qual a estratégia da mineradora para atender ao pedido, de forma que as especificações de qualidade estejam mais próximas das metas especificadas? Observação: considere que a penalidade pelo desvio de atendimento à meta é igual ao produto do peso de comparação pelo correspondente peso do critério.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$parametros$	:	Conjunto de parâmetros de controle;
$pilhas$	:	Conjunto de pilhas;
tl_j	:	Teor mínimo admissível para o parâmetro j no produto final (%);
tr_j	:	Teor desejável para o parâmetro j no produto final (%);
wdt_j	:	Peso do desvio da meta para o parâmetro j;
dtn_j	:	Desvio negativo da meta para o parâmetro j, em toneladas;
dtp_j	:	Desvio positivo da meta para o parâmetro j, em toneladas;
Qu_i	:	Quantidade máxima disponível na pilha i, em toneladas;
$nunidret_i$	:	Número de unidades de retomada;
t_{ij}	:	Teor do parâmetro j na pilha i (%);
p	:	Produção total requerida, em toneladas.

e as seguintes variáveis de decisão:

x_i : Quantidade de minério a ser retirada da pilha i, em toneladas;

y_i : $\begin{cases} 1 & \text{se a pilha } i \text{ for usada;} \\ 0 & \text{caso contrário.} \end{cases}$

O modelo de programação matemática para o Exercício 2.19 é:

$$\min \sum_{j \in parametros} (wdt_j \times dtn_j + wdt_j \times dtp_j)$$

$$\begin{aligned}
\sum_{i \in pilhas} (t_{ij} - tu_j)x_i &\leq 0 && \forall j \in parametros \\
\sum_{i \in pilhas} (t_{ij} - tl_j)x_i &\geq 0 && \forall j \in parametros \\
\sum_{i \in pilhas} ((t_{ij} - tr_j) \times x_i) + dtn_j - dtp_j &= 0 && \forall j \in parametros \\
x_i &\leq Qu_i && \forall i \in pilhas \\
\sum_{i \in pilhas} x_i &= p && \\
nunidret_i &= x_i/unidret && \forall i \in pilhas \\
y_i &\geq x_i/Qu_i && \forall i \in pilhas \mid Qu_i \neq 0 \\
x_i &\geq retmin \times y_i && \forall i \in pilhas \\
nunidret_i &\in \mathbb{Z}^+ && \forall i \in pilhas \\
y_i &\in \{0,1\} && \forall i \in pilhas
\end{aligned}$$

Seguem a planilha com os parâmetros de entrada e saída para o Problema da Mistura em questão, juntamente com a implementação LINGO.

Modelo Lingo

MISTURA DE MINÉRIOS COM METAS

Produção: 6.000 ← p

retmin

Retomada Min: 500

Unid.de Retomada: 10

unidret

	Fe (%)	Al2O3 (%)	P (%)	PPC (%)	He (%)	
Limite Superior	49,5	0,37	0,043	2,65	50	← tu
Meta	47,0	0,32	0,040	2,35	40	← tr
Limite Inferior	44,5	0,27	0,035	2,05	38	← tl
						← parametros
Peso Comparação	1	100	1000	10	1	
Critério	5	0	100	10	1	
Peso Desvio da Meta	5	0	100000	100	0	← wdt
Mistura	47,0	0,29	0,040	2,35	45	
Desvio negativo (100t)	0,00	208,60	0,00	0,00	0,00	← dtn
Desvio positivo (100t)	0,50	0,00	0,00	0,00	31060,00	← dtp

Critérios	Pesos
Irrelevante	0
Importante	1
Muito Importante	5
Crítico	10
Muito Crítico	100

Custo Total: R$ 70.759,00

solucao

	Massa (t)	Fe (%)	Al2O3 (%)	P (%)	PPC (%)	He (%)	Custos ($/t)	A retomar (t)
Pilha 1	1500	52,64	0,52	0,084	4,48	45	10,50	0,00
Pilha 2	2000	39,92	0,18	0,029	0,65	97	12,50	500,00
Pilha 3	1700	47,19	0,50	0,050	2,52	52	12,00	1480,00
Pilha 4	1450	49,36	0,22	0,039	1,74	78	10,00	980,00
Pilha 5	1250	43,94	0,46	0,032	2,36	41	11,50	0,00
Pilha 6	1890	48,97	0,54	0,057	4,34	90	11,00	0,00
Pilha 7	1640	47,46	0,20	0,047	5,07	9	10,80	0,00
Pilha 8	1124	46,52	0,32	0,039	3,51	4	11,20	0,00
Pilha 9	1990	56,09	0,95	0,059	4,10	80	10,40	0,00
Pilha 10	900	46,00	0,26	0,031	2,51	21	12,00	730,00
Pilha 11	1540	49,09	0,22	0,040	4,20	12	10,30	0,00
Pilha 12	1630	49,77	0,20	0,047	4,81	12	11,90	0,00
Pilha 13	1320	53,03	0,24	0,047	4,17	1	12,30	1190,00
Pilha 14	1245	52,96	0,29	0,052	4,81	1	11,10	0,00
Pilha 15	1859	42,09	0,17	0,031	1,38	47	12,10	1120,00

pilhas Qu teor custo

```
title: MisturaMetas(R).lg4;

sets:
  parametros /@ole('MisturaMetas(R).xls','parametros')/: tl, tu, tr, wdt, dtn, dtp;
  pilhas /@ole('MisturaMetas(R).xls','pilhas')/: Qu, nunidret, x, y;
  matriz(pilhas,parametros): t;
endsets

data:
  ! Importa os dados do Excel;
  tl, tu, tr, t, wdt, Qu, p, retmin, unidret =
      @ole('MisturaMetas(R).xls','tl','tu','tr','teor','wdt','Qu','p','retmin','unidret');
enddata
! Minimizar o desvio do teor de cada parâmetro j em relação a sua meta de qualidade;
[fo] min = @sum(parametros(j): wdt(j)*dtn(j) + wdt(j)*dtp(j));

! O limite superior de especificação deve ser satisfeito para cada parâmetro j;
@for(parametros(j): @sum(pilhas(i): (t(i,j) - tu(j))*x(i)) <= 0);
! O limite inferior de especificação deve ser satisfeito para cada parâmetro j;
@for(parametros(j): @sum(pilhas(i): (t(i,j) - tl(j)) × x(i)) >= 0);
! A meta de qualidade deve ser buscada para cada parâmetro j;
@for(parametros(j): @sum(pilhas(i): (t(i,j) - tr(j)) × x(i)) + dtn(j) - dtp(j) = 0);
```

```
! A quantidade a ser retomada em cada pilha i deve ser inferior ou igual a Qu(i);
@for(pilhas(i): @BND(0, x(i), Qu(i)));

! A produção total deve ser igual a p;
@sum(pilhas(i): x(i)) = p;

! A quantidade x(i) a ser retomada na pilha i deve ser múltipla de unidret;
@for(pilhas(i): nunidret(i) = x(i) / unidret);

! Se for retomada qualquer quantidade na pilha i então y(i) = 1.
  Caso contrario, y(i) assume valor 0;
@for(pilhas(i) | Qu(i) #ne# 0: y(i) >= x(i)/Qu(i));

! Se for retomar alguma pilha i a quantidade x(i) a retomar deve ser superior ou
  igual a retmin;
@for(pilhas(i): x(i) >= retmin*y(i));
! A variável nunidred(i) é inteira e y(i) binária;
@for(pilhas(i):
  @GIN(nunidret(i));
  @BIN(y(i));
);

data:
  ! Exporta os resultados para Excel;
  @ole('MisturaMetas(R).xls','dtn','dtp','solucao') = dtn, dtp, x;
enddata
```

2.20 Problema das Usinas

Uma empresa siderúrgica possui 3 usinas e cada uma delas requer uma quantidade mensal mínima de minério para operar. A empresa adquire minério de 4 minas diferentes. Cada uma das minas tem uma capacidade máxima de produção mensal estabelecida. Por imposições contratuais, o custo do minério para a empresa é composto por um custo fixo mensal para cada mina (este valor é pago em caso de haver produção na mina), mais um custo de transporte ($/t) que varia de acordo com a distância entre as minas e usinas (cada par mina/usina tem um custo diferente). Os dados são mostrados na tabela a seguir:

MINAS	Usina 1	Usina 2	Usina 3	Cap. máx. das minas (t/mês)	Custo Fixo ($)
Mina 1 ($/t)	10	8	13	11500	50000
Mina 2 ($/t)	7	9	14	14500	40000
Mina 3 ($/t)	6,5	10,8	12,4	13000	30000
Mina 4 ($/t)	8,5	12,7	9,8	12300	25500
Quant. req. (t/mês)	10000	15400	13300	-	-

Construir um modelo de otimização para determinar a quantidade de minério a ser comprada de cada mina e levada a cada usina de forma a minimizar o custo total de compra de minério.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$minas$: Conjunto de Minas ;
$usinas$: Conjunto de usinas;
cap_i : Capacidade de produção da mina i;
$cfixo_i$: Custo fixo de utilização da mina i;
$demanda_j$: Demanda requerida pela usina j;
$custo_{ij}$: Custo de transporte de minério da mina i para a usina j.

e as seguintes variáveis de decisão:

x_{ij} : Quantidade de minério a ser transportado da mina i para a usina (j), em toneladas;
y_i : $\begin{cases} 1 & \text{se a mina } i \text{ for usada;} \\ 0 & \text{caso contrário.} \end{cases}$

O modelo de programação matemática para o Exercício 2.20 é:

$$\min \sum_{i \in minas} \sum_{j \in usinas} (custo_{ij} \times x_{ij}) + \sum_{i \in minas} (cfixo_i \times y_i)$$

s.a:

$$\sum_{j \in usinas} x_{ij} \leq cap_i \qquad \forall i \in minas$$

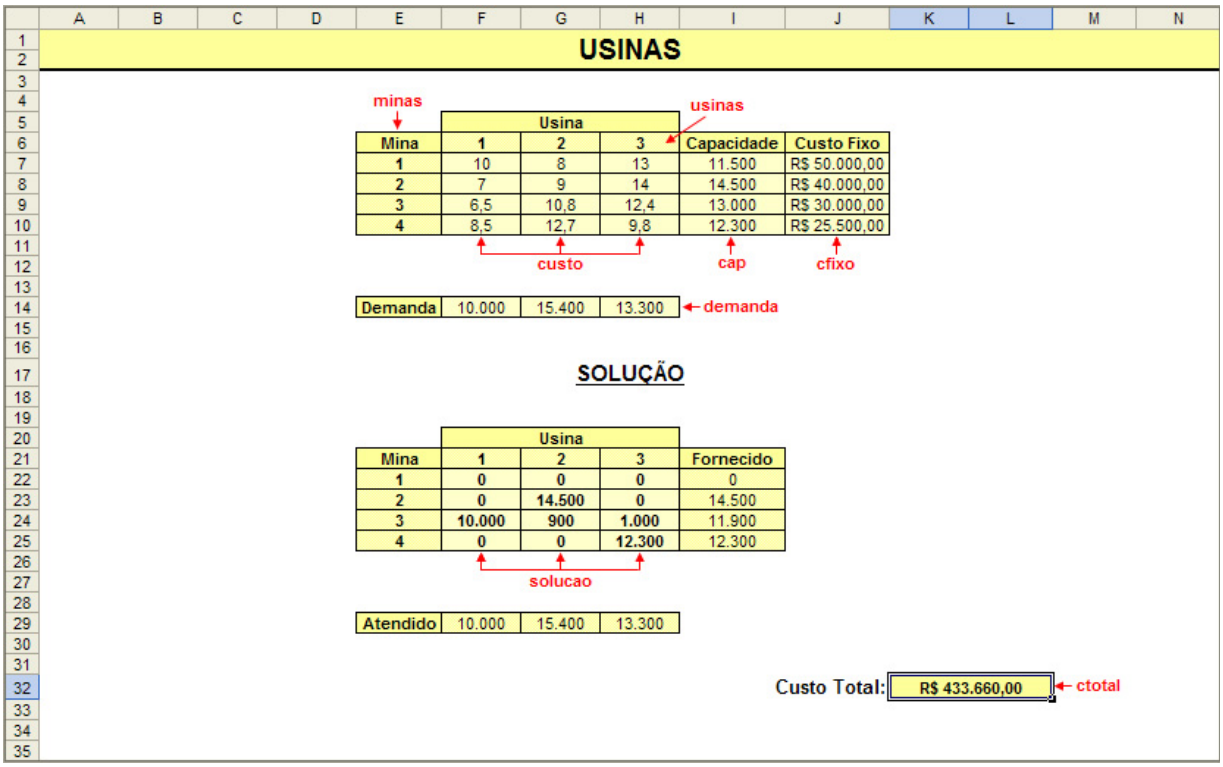
$$\sum_{i \in minas} x_{ij} = demanda_j \qquad \forall j \in usinas$$

$$y_i \geq (\sum_{j \in usinas} x_{ij})/cap_i \qquad \forall i \in minas$$

$$x_{ij} \geq 0 \qquad \forall i \in minas, \forall j \in usinas$$

$$y_i \in \{0, 1\} \qquad \forall i \in minas$$

Modelo LINGO



USINAS

Mina	Usina 1	Usina 2	Usina 3	Capacidade	Custo Fixo
1	10	8	13	11.500	R$ 50.000,00
2	7	9	14	14.500	R$ 40.000,00
3	6,5	10,8	12,4	13.000	R$ 30.000,00
4	8,5	12,7	9,8	12.300	R$ 25.500,00

minas, usinas, custo, cap, cfixo

Demanda	10.000	15.400	13.300

demanda

SOLUÇÃO

Mina	Usina 1	Usina 2	Usina 3	Fornecido
1	0	0	0	0
2	0	14.500	0	14.500
3	10.000	900	1.000	11.900
4	0	0	12.300	12.300

solucao

Atendido	10.000	15.400	13.300

Custo Total: R$ 433.660,00 ← ctotal

```
Title: Usinas(R).lg4;

sets:
  minas/@ole('Usinas(R).xls','minas')/: cap, cfixo, y;
  usinas/@ole('Usinas(R).xls','usinas')/: demanda;
  matriz(minas,usinas): custo, x;
endsets

data:
  ! Importa os dados do Excel;
  cap, cfixo, demanda, custo = @ole('Usinas(R).xls','cap','cfixo','demanda','custo');
enddata

! Minimiza os custos com transporte entre as minas e o custo fixo de utilização das minas;
[fo] min = @sum(matriz(i,j): custo(i,j)*x(i,j)) + @sum(minas(i): cfixo(i)*y(i));
! Total transportado de uma mina para as usinas deve ser menor ou igual à
  capacidade de produção da mina;
@for(minas(i): @sum(usinas(j): x(i,j)) <= cap(i));

! A quantidade de minério que chega a uma usina deve ser igual à demanda da mesma,
  uma vez que a oferta é maior que a demanda ;
@for(usinas(j):@sum(minas(i): x(i,j)) = demanda(j));
! Se houver produção na mina i então y(i) = 1;
@for(minas(i): y(i) >= @sum(usinas(j): x(i,j)) / cap(i));
```

```
! As variáveis y(i) são binárias;
@for(minas(i): @BIN(y(i)));

data:
 ! Exporta os dados para o Excel;
 @ole('Usinas(R).xls','solucao','ctotal') = x, fo;
enddata
```

2.21 Dimensionamento de lotes

Empresas de manufatura fabricam, em geral, diversos tipos de produtos solicitados por diferentes clientes, muitas vezes em grandes quantidades, os quais devem estar prontos para entrega em diferentes datas previamente agendadas. Como as fábricas têm capacidade de produção limitada devido à quantidade de máquinas, mão-de-obra etc. disponíveis, é necessário planejar a produção, isto é, decidir o *quê* e *quanto* produzir (em outras palavras, dimensionar os lotes de produção), assim como *quando* produzir. A necessidade de antecipação da fabricação de produtos estocados de um período para outro acarreta custos de estocagem e algumas dificuldades operacionais. No planejamento da produção, deseja-se determinar o tamanho dos lotes de produção para atender a demanda na data solicitada de modo que a soma dos custos de produção e estocagem seja mínima.

Considere uma empresa que fabrica n produtos e deseja programar sua produção nos próximos $|T|$ períodos de tempo. Este conjunto de períodos de tempo para o qual a empresa planeja sua produção é denominado *horizonte de planejamento*. Suponhamos que a demanda de cada produto em cada período do horizonte de planejamento é conhecida. Em cada período, os recursos necessários para a produção são limitados e renováveis, isto é, em cada período, uma quantidade de recursos está disponível e não depende de como foram utilizados nos períodos anteriores. Exemplos de recursos renováveis são mão-de-obra, energia elétrica e horas de máquinas, enquanto recursos não renováveis são, por exemplo, matérias-primas que sobram em um período e podem ser utilizadas nos períodos seguintes. Há a possibilidade de estocagem de produtos de um período para o outro. Considere os seguintes dados do problema:

d_{it} : demanda do item i no período t
R_t : disponibilidade de recursos renováveis no período t
r_i : quantidade de recursos necessários para a produção de uma unidade do item i
c_{it} : custo de produzir uma unidade do item i no período t
h_{it} : custo de estocar uma unidade do item i no período t
T : conjunto dos períodos do horizonte de planejamento

e as seguintes variáveis de decisão:

x_{it} : número de itens do tipo i produzidos no período t
I_{it} : número de itens do tipo i em estoque no final do período t

Os estoques iniciais do horizonte de planejamento I_{i0} são dados. A seguir, são detalhadas as restrições do problema.

$$\text{minimizar} \quad \sum_{i=1}^{n}\sum_{t=1}^{|T|}(c_{it}x_{it} + h_{it}I_{it}) \tag{2.1}$$

$$x_{i,t} + I_{i,t-1} - I_{it} = d_{i,t} \quad \forall\, i = 1, 2, \ldots, n;\ t = 1, 2, \cdots, |T| \tag{2.2}$$

$$\sum_{i=1}^{n} r_i x_{it} \leq R_t \quad \forall\, t = 1, 2, \cdots, |T| \tag{2.3}$$

$$I_{it} \in \mathbb{Z}^+ \quad \forall\, i = 1, 2, \cdots, n;\ t = 1, 2, \cdots, |T| \tag{2.4}$$

$$x_{it} \in \mathbb{Z}^+ \quad \forall\, i = 1, 2, \cdots, n;\ t = 1, 2, \cdots, |T| \tag{2.5}$$

A função objetivo (2.1) visa a minimização dos custos com a produção e a estocagem dos produtos em cada período do horizonte de planejamento. As restrições (2.2) são de conservação de estoque. Assim, para cada item i, o nível de estoque no final do período t é igual ao que se tinha em estoque no final do período anterior $(t-1)$, adicionado ao montante que foi produzido no período t, menos o que foi demandado no período t. As restrições (2.3) asseguram que a capacidade requerida para a produção dos itens em cada período t não pode superar a capacidade disponível da fábrica. As restrições (2.4) asseguram o atendimento à demanda. De fato, a quantidade x_{it} do item i produzida no período t mais a quantidade em estoque no final do período anterior, $I_{i,t-1}$, deve ser maior ou igual a d_{it}, ou seja, $x_{it} + I_{i,t-1} \geq d_{it}$. Nesta última expressão, o termo à esquerda representa, por definição, I_{it}. Logo, garante-se o atendimento às demandas impondo-se $I_{it} \geq 0$ e inteiro.

2.22 Planejamento da produção

[Retirado de Arenales *et al.* (2007). Vide [1]] A seguir, são apresentados alguns modelos importantes de planejamento da produção, conhecidos na literatura como modelos de dimensionamento de lotes (*lot sizing*). Os modelos apresentados possuem as seguintes características comuns: (a) o horizonte de planejamento é finito e dividido em períodos; (b) a demanda de cada item em cada período é dinâmica, isto é, varia ao longo do horizonte de planejamento; (c) a demanda e outros parâmetros dos modelos são supostos conhecidos, isto é, tratam-se de modelos determinísticos. Existem outros modelos em que o horizonte pode ser infinito, a variável tempo é contínua, a demanda é expressa como uma taxa em relação ao tempo, e ainda modelos em que a demanda é estocástica. Para enunciar os modelos matemáticos, considere os seguintes parâmetros:

d_{it} : demanda do item i no período t
b_i : tempo para produzir uma unidade do item i
C_t : capacidade de produção, em horas, de uma máquina ou instalação no período t
sp_i : tempo de preparação de máquina para processar o item i
s_i : custo de preparação do item i
h_i : custo unitário de estoque do item i
I_i^0 : estoque inicial do item i
n : número de itens finais
T : conjunto dos períodos do horizonte de planejamento

Considere, também, as seguintes variáveis de decisão:
x_{it} = quantidade do item i produzida no período t (tamanho do lote)
I_{it} = estoque do item i no fim do período t
$y_{it} = \begin{cases} 1, & \text{se o item } i \text{ é produzido no período } t \\ 0, & \text{caso contrário.} \end{cases}$

2.22.1 Um item sem restrição de capacidade

[Retirado de Arenales *et al.* (2007). Vide [1]] O problema mais simples de dimensionamento de lotes envolve um único item, sem restrição de capacidade. Neste caso, h é o custo unitário de estoque; s é o custo de preparação do item; I_0 é o estoque inicial; I_t é o estoque do item no fim do período t; x_t é a quantidade do item produzida no período t e $y_t = 1$ se o item for produzido no período t e 0, caso contrário. Seu modelo é:

$$\min \quad \sum_{t \in T} (s \times y_t + h \times I_t^+) \qquad \text{(DL1)}$$

$$I_t = I_{t-1} + x_t - d_t \quad \forall t \in T, \quad I_0 = I_{|T|} = 0 \qquad \text{(DL2)}$$

$$x_t \leq \left(\sum_{k=t}^{|T|} d_k \right) y_t \quad \forall t \in T \qquad \text{(DL3)}$$

$$x_t \geq 0 \quad \forall t \in T \qquad \text{(DL4)}$$

$$I_t \geq 0 \quad \forall t \in T \qquad \text{(DL5)}$$

$$y_t \in \{0, 1\} \quad \forall t \in T \qquad \text{(DL6)}$$

A função objetivo (DL1) minimiza o custo total de preparação e estoque. As restrições (DL2) representam equações de balanceamento de estoque em cada período t. Se $I_0 > 0$ use esse estoque inicial para abater demandas no horizonte; portanto, pode-se assumir, sem perda de generalidade, que $I_0 = 0$. Além disso, na solução ótima $I_{|T|} = 0$ é uma decorrência da minimização do custo de estoque. As restrições (DL3) garantem que a produção no período t é limitada superiormente pela demanda acumulada do período t ao último período $|T|$, e que o tamanho do lote é positivo, isto é, $x_t > 0$ somente se há produção no período t ($y_t = 1$). As restrições (DL4), (DL5) e (DL6) indicam o tipo das variáveis.

No caso em que a demanda pode ser atendida com atraso, atribui-se uma penalidade δ por unidade de demanda não atendida no período t. Considere as variáveis:

I_t^+ : estoque no fim do período t
I_t^- : falta (demanda não atendida) no período t

Com a introdução destas novas variáveis, a formulação anterior é modificada para a seguinte:

$$\min \quad \sum_{t \in T} (s \times y_t + h \times I_t + \delta \times I_t^-) \qquad \text{(DL7)}$$

$$I_t^+ - I_t^- = I_{t-1}^+ - I_{t-1}^- + x_t - d_t \quad \forall t \in T, \quad I_0^+ = I_0^- = 0 \qquad \text{(DL8)}$$

$$x_t \leq \left(\sum_{k=t}^{|T|} d_k \right) y_t \quad \forall t \in T \qquad \text{(DL9)}$$

$$x_t \geq 0 \quad \forall t \in T \qquad \text{(DL10)}$$

$$I_t^+ \geq 0 \quad \forall t \in T \qquad \text{(DL11)}$$

$$I_t^- \geq 0 \quad \forall t \in T \qquad \text{(DL12)}$$

$$y_t \in \{0, 1\} \quad \forall t \in T \qquad \text{(DL13)}$$

A função objetivo (DL7) minimiza o custo total de preparação, estoque e demanda não atendida. As restrições de balanceamento (DL8) levam em consideração o estoque e a demanda não atendida em cada período t. As restrições (DL9) são idênticas às restrições (DL3), e as restrições (DL10) a (DL13) indicam o tipo das variáveis.

A consideração de demanda não atendida pode ser incluída nos modelos a seguir, de acordo com o exposto anteriormente.

2.22.2 Múltiplos itens e restrição de capacidade

[Retirado de Arenales *et al.* (2007). Vide [1]] Este problema, conhecido na literatura como *capacitated lot sizing problem*, trata de um conjunto N de itens que devem ser processados em uma única máquina ou facilidade, com restrições de capacidade, e pode ser modelado como:

$$\min \quad \sum_{i\in N}\sum_{t\in T}(s_i \times y_{it} + h_i \times I_{it}) \tag{DL14}$$

$$I_{it} = I_{i,t-1} + x_{it} - d_{it} \qquad \forall i \in N,\ \forall t \in T, \quad I_{i0} = 0 \tag{DL15}$$

$$\sum_{i\in N}(sp_i \times y_{it} + b_i \times x_{it}) \leq C_t \qquad \forall t \in T \tag{DL16}$$

$$x_{it} \leq M_{it} \times y_{it} \qquad \forall t \in T \tag{DL17a}$$

$$M_{it} = \min\{\tfrac{C_t - sp_i}{b_i}, \sum_{k=t}^{|T|} d_{ik}\} \qquad \forall i \in N,\ \forall t \in T \tag{DL17b}$$

$$x_{it} \geq 0 \qquad \forall i \in N,\ \forall t \in T \tag{DL18}$$

$$I_{it} \geq 0 \qquad \forall i \in N,\ \forall t \in T \tag{DL19}$$

$$y_{it} \in \{0,1\} \qquad \forall i \in N,\ \forall t \in T \tag{DL20}$$

A função objetivo (DL14) minimiza o custo total de preparação e estoque. As restrições (DL15) correspondem ao balanceamento de estoque de cada item i em cada período t. As restrições (DL16) expressam que em cada período t, o tempo total de preparação e produção é limitado pela capacidade disponível. As restrições (DL17a) implicam que $x_{it} > 0$ se e somente se $y_{it} = 1$. O limitante M_{it} em (DL17b) é o mínimo entre a capacidade restante no período t (se i é produzido nesse período) e a demanda acumulada do período t ao período $|T|$. As demais restrições indicam o tipo das variáveis de decisão.

2.23 Representação de restrições disjuntivas

Em muitos problemas práticos, requer-se que apenas uma dentre várias restrições seja satisfeita. Esse tipo de situação aparece com frequência em problemas de programação de tarefas em máquinas, como os apresentados na Seção 2.24.

Consideremos inicialmente o caso de duas restrições, sendo que apenas uma delas deve estar ativa. Sejam as restrições:

$$r_1(x_1, x_2, \cdots, x_n) \leq b_1 \tag{2.6}$$

$$r_2(x_1, x_2, \cdots, x_n) \leq b_2 \tag{2.7}$$

Neste caso, basta definir uma variável binária y tal que se $y = 1$ então r_1 estará ativa e se $y = 0$ então é r_2 quem estará ativa. Matematicamente, podemos representar isto por:

$$r_1(x_1, x_2, \cdots, x_n) \leq b_1 + M(1 - y) \tag{2.8}$$

$$r_2(x_1, x_2, \cdots, x_n) \leq b_2 + My \tag{2.9}$$

em que M é um número suficientemente grande.

Exemplo: Considere as restrições:

$$4x_1 + 2x_2 \leq 80 \tag{2.10}$$

$$2x_1 + 5x_2 \leq 100 \tag{2.11}$$

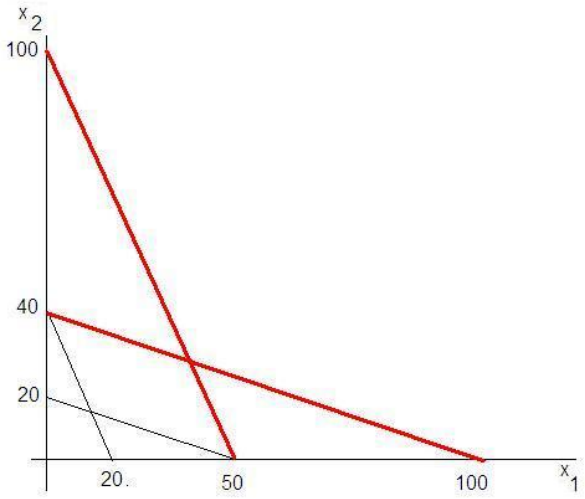


Figura 3: Exemplo de restrições disjuntivas

As linhas cheias da Figura 3 mostram as retas que geram os semi-espaços definidos pelas restrições (2.10) e (2.11). Já as linhas vermelhas (mais grossas) mostram o quanto essas retas têm que ser transladadas de forma que elas sejam redundantes, isto é, não ativas.

Para encontrar as retas vermelhas (em linhas mais grossas) proceda como segue. Gere, inicialmente, a reta base. Seja a reta $4x_1 + 2x_2 = 80$, a qual é geradora do semi-espaço definido pela restrição (2.10). Explicitando x_2 dessa equação e simplificando, tem-se: $2x_2 = 80 - 4x_1$. Para ser redundante, essa reta tem que ser transladada, no mínimo, 30 unidades à direita do ponto (20, 0), isto é, deve passar pelo ponto (50, 0) ou à direita dele pois, caso contrário, ela interceptaria o semi-espaço definido pela restrição $2x_1 + 5x_2 \leq 100$, eliminando alguns pontos do espaço de soluções viáveis. Assim, precisamos determinar o valor de M tal que $2x_2 = 80 + M - 4x_1$ passe pelo ponto (50, 0), isto é, $0 = 80 + M - 4 \times 50$. Resolvendo, tem-se $M = 120$. De forma análoga, repetindo-se o procedimento para a outra reta, conclui-se que o valor M a ser adicionado ao lado direito da segunda restrição é 100.

Observe que o menor valor de M é dado por $\max\{120, 100\} = 120$. Um valor pequeno de M é desejável para acelerar os métodos de resolução de problemas de programação inteira. A representação matemática da disjunção das duas restrições apresentadas é dada, então, por:

$$4x_1 + 2x_2 \leq 80 + 120(1 - y) \tag{2.12}$$
$$2x_1 + 5x_2 \leq 100 + 120y \tag{2.13}$$

Se tivermos m restrições $r_1(x) \leq b_1$, $r_2(x) \leq b_2$, $\cdots$, $r_m(x) \leq b_m$ e apenas uma delas deve estar ativa ao mesmo tempo, então criamos m variáveis binárias y_i e escrevemos:

$$r_1(x_1, x_2, \cdots, x_n) \leq b_1 + M(1 - y_1) \tag{2.14}$$
$$r_2(x_1, x_2, \cdots, x_n) \leq b_2 + M(1 - y_2) \tag{2.15}$$
$$\vdots \tag{2.16}$$
$$r_i(x_1, x_2, \cdots, x_n) \leq b_i + M(1 - y_i) \tag{2.17}$$
$$\vdots \tag{2.18}$$
$$r_m(x_1, x_2, \cdots, x_n) \leq b_n + M(1 - y_m) \tag{2.19}$$
$$y_1 + y_2 + \cdots + y_i + \cdots + y_m = 1 \tag{2.20}$$
$$y_i \in \{0, 1\} \quad \forall i = 1, 2, \cdots, m \tag{2.21}$$

Assim, se $y_i = 1$ então $r_i(x) \leq b_i + M(1 - y_i) = b_i$ e esta restrição fica ativa e as demais, inativas. Se $y_i = 0$ então $r_i(x) \leq b_i + M(1 - y_i) = b_i + M$ fica redundante (inativa) e, portanto, satisfeita para qualquer valor de x, uma vez que M é um número arbitrariamente grande.

2.24 Sequenciamento em uma máquina

[Retirado de Arenales *et al.* (2007) [1]] Considere um conjunto N de tarefas a serem processadas em uma máquina. Todas as tarefas estão disponíveis para processamento no instante zero e admite-se que a interrupção (*preemption*) de qualquer tarefa não é permitida. Considere os seguintes parâmetros inteiros e não-negativos:

p_i : tempo de processamento da tarefa i
d_i : data de entrega da tarefa i
M : número arbitrariamente grande

Sejam as seguintes variáveis de decisão:

C_i = instante de término do processamento da tarefa i
T_i = atraso da tarefa i, dado por $T_i = \max\{C_i - d_i, 0\}$
E_i = adiantamento ou avanço da tarefa i, dado por $E_i = \max\{d_i - C_i, 0\}$
L_i = *lateness* da tarefa i, isto é, $L_i = C_i - d_i$

$$x_{ij} = \begin{cases} 1, & \text{se a tarefa } i \text{ precede imediatamente a tarefa } j \\ 0, & \text{caso contrário.} \end{cases}$$

Observe que o *lateness* mede o grau de atraso, o qual pode ser negativo. Neste último caso, o valor negativo indica que a tarefa não está atrasada e, sim, adiantada em relação à sua data de entrega.

Seja zero (0) uma tarefa fictícia que precede imediatamente a primeira tarefa e sucede imediatamente a última tarefa de uma sequência de tarefas. A partir desses parâmetros e variáveis, é possível formular problemas com critérios distintos de otimização. As seguintes restrições são comuns a todos os problemas:

$$\sum_{i \in N \cup \{0\}, i \neq j} x_{ij} = 1 \qquad \forall j \in N \cup \{0\} \tag{S1}$$
$$\sum_{j \in N \cup \{0\}, j \neq i} x_{ij} = 1 \qquad \forall i \in N \cup \{0\} \tag{S2}$$
$$C_j \geq C_i - M + (p_j + M)x_{ij} \qquad \forall i \in N \cup \{0\},\ \forall j \in N \tag{S3}$$
$$C_i \geq 0 \qquad \forall i \in N,\ C_0 = 0 \tag{S4}$$
$$x_{ij} \in \{0, 1\} \qquad \forall i \in N \cup \{0\},\ \forall j \in N \cup \{0\} \tag{S5}$$

As restrições (S1) e (S2) garantem que cada tarefa tem apenas uma tarefa imediatamente predecessora e uma tarefa imediatamente sucessora, respectivamente. Quando $x_{ij} = 1$ a restrição (S3) implica que $C_j \geq C_i + p_j$ e se $x_{ij} = 0$, então $C_j - C_i \geq -M$, isto é, a restrição (S3) fica redundante (desativada). As restrições (S4) e (S5) indicam o tipo das variáveis. Observe que C_0 é fixado em ZERO.

2.24.1 Minimização do tempo de fluxo total

O tempo de fluxo total corresponde à soma dos tempos de término das tarefas, isto é, $\sum_{i \in N} C_i$. O problema, então, consiste em:

$$\min \sum_{i \in N} C_i$$
$$\text{(S1) - (S5)}$$

Pode-se demonstrar que a solução ótima deste problema é dada pela regra SPT (*shortest processing time*), em que as tarefas são sequenciadas em ordem não decrescente dos tempos de processamento, isto é, as tarefas são processadas na sequência $[1], [2], ..., [n]$, tal que $p_{[1]} \leq p_{[2]} \leq \cdots \leq p_{[n]}$ e $[i]$ corresponde à tarefa da i-ésima posição. Por exemplo, $p_{[2]}$ é o tempo de processamento da tarefa que ocupa a segunda posição na sequência de produção.

Exercício:
Determine o tempo de fluxo total do problema de sequenciamento a seguir, em que há 7 tarefas com os seguintes tempos de processamento e datas de entrega.

Tarefas	1	2	3	4	5	6	7
Tempo de processamento (p)	7	4	2	5	6	3	1
Data de entrega (d)	13	10	6	9	3	20	4

Segue o modelo LINGO. Observe, neste modelo, que o conjunto *Tarefas* envolve as tarefas reais $1, 2, \cdots, 7$ e mais a tarefa fictícia 0. Assim, quando é necessário referenciar apenas as tarefas reais, é necessário excluir a tarefa 0 (de índice 1).

```
sets:
  Tarefas / 0 1 2 3 4 5 6 7/: p, d, C;
  Matriz(Tarefas, Tarefas): x;
endsets

data:
  p = 0  7  4  2  5  6  3 1;
  d = 0 13 10  6  9  3 20 4;
  M = 1000;
enddata

[fo] min = @sum(Tarefas(i) | i #NE# 1: C(i));

@for(Tarefas(j):
   @sum(Tarefas(i) | i #NE# j: x(i,j)) = 1);

@for(Tarefas(i):
```

```
   @sum(Tarefas(j) | j #NE# i: x(i,j)) = 1);

@for(Tarefas(j) | j #NE# 1:
   @for(Tarefas(i):
     C(j) >= C(i) - M + (p(j) + M)*x(i,j)));

C(1) = 0;

@for(Tarefas(j):
   @for(Tarefas(i):
     @bin(x(i,j))));
```

O fluxo total ótimo deste exemplo é 84, sendo a sequência ótima dada por $7 \to 3 \to 6 \to 2 \to 4 \to 5 \to 1$. As datas de término são $C(1) = 28$, $C(2) = 10$, $C(3) = 3$, $C(4) = 15$, $C(5) = 21$, $C(6) = 6$ e $C(7) = 1$.

2.24.2 Minimização do atraso máximo

O atraso máximo, denotado por T_{max}, está associado à tarefa com maior diferença entre seu instante de término e data de entrega, isto é, $T_{max} = \max_{i \in N} T_i$. Observe que $T_i = \max\{C_i - d_i, 0\}$ implica $T_i \geq C_i - d_i$ e $T_i \geq 0$. O problema de minimização de T_{max} pode, então, ser formulado como:

$$\begin{array}{lrcll} \min & T_{max} & & & \\ & T_{max} & \geq & T_i & \forall i \in N \\ & T_i & \geq & C_i - d_i & \forall i \in N \\ & T_i & \geq & 0 & \forall i \in N \\ & \text{(S1)} & - & \text{(S5)} & \end{array}$$

Demonstra-se que a solução ótima deste problema é dada pela regra EDD (*Earliest Due Date*), que consiste em sequenciar as tarefas em ordem não decrescente das datas de entrega, isto é, as tarefas são processadas na sequência: $[1], [2], \cdots, [n]$ tal que $d_{[1]} \leq d_{[2]} \leq \cdots \leq d_{[n]}$.

2.24.3 Minimização da soma dos atrasos

Com este critério de otimização, o problema é modelado como:

$$\begin{array}{lrcll} \min & \sum_{i \in N} T_i & & & \\ & T_i & \geq & C_i - d_i & \forall i \in N \\ & T_i & \geq & 0 & \forall i \in N \\ & \text{(S1)} & - & \text{(S5)} & \end{array}$$

2.24.4 Minimização da soma dos atrasos e adiantamentos

Este problema é modelado como:

$$\begin{array}{lrcll} \min & \sum\limits_{i \in N} (T_i + E_i) & & & \\ & T_i & \geq & C_i - d_i & \forall i \in N \\ & E_i & \geq & d_i - C_i & \forall i \in N \\ & T_i & \geq & 0 & \forall i \in N \\ & E_i & \geq & 0 & \forall i \in N \\ & \text{(S1)} & - & \text{(S5)} & \end{array}$$

2.24.5 Minimização do número de tarefas atrasadas

Sejam as variáveis $y_i = 1$ se a tarefa i está atrasada e $y_i = 0$, caso contrário. Então a formulação do problema é dada por:

$$\begin{array}{lrcll} \min & \sum\limits_{i \in N} y_i & & & \\ & T_i & \geq & C_i - d_i & \forall i \in N \\ & T_i & \leq & M y_i & \forall i \in N \\ & T_i & \geq & 0 & \forall i \in N \\ & y_i & \in & \{0, 1\} & \forall i \in N \\ & \text{(S1)} & - & \text{(S5)} & \end{array}$$

Note que se $T_i > 0$ então $y_i = 1$.

2.24.6 Minimização do *lateness* máximo

Seja $L_{max} = \max\limits_{i \in N} L_i$ o *lateness* máximo. Como a variável L_i é livre, isto é, pode assumir valores negativos ou nulos ou positivos, definamos $L_i = L_i^+ - L_i^-$, com $L_i^+ \geq 0$ e $L_i^- \geq 0$. Desta maneira, o problema pode ser representado pelo seguinte modelo, onde todas as variáveis são não-negativas:

$$\begin{array}{lrcll} \min & L_{max} & & & \\ & L_{max} & \geq & L_i^+ - L_i^- & \forall i \in N \\ & L_i^+ - L_i^- & = & C_i - d_i & \forall i \in N \\ & L_i^+ & \geq & 0 & \forall i \in N \\ & L_i^- & \geq & 0 & \forall i \in N \\ & \text{(S1)} & - & \text{(S5)} & \end{array}$$

Demonstra-se que a solução ótima deste problema também é dada pela regra EDD (*Earliest Due Date*).

2.24.7 Sequenciamento com tempo de preparação de máquina

Considere agora que para processar uma tarefa i seja necessário um tempo de preparação de máquina dado por s_i, conhecido como tempo de *setup*. Se esse tempo for independente da sequência, basta incorporá-lo ao tempo de execução dessa tarefa, isto é, basta considerar o tempo de processamento da tarefa i como $s_i + p_i$. Neste caso, o conjunto de restrições (S3) é alterado para:

$$C_j \quad \geq \quad C_i - M + (s_j + p_j + M) x_{ij} \quad \forall i \in N \cup \{0\},\ \forall j \in N \quad \text{(S6)}$$

Se, por outro lado, o tempo de preparação for dependente da sequência, então o conjunto de restrições (S3) deve ser substituído por:

$$C_j \quad \geq \quad C_i - M + (s_{ij} + p_j + M) x_{ij} \quad \forall i \in N \cup \{0\},\ \forall j \in N \quad \text{(S7)}$$

onde s_{ij} indica o tempo de preparação da máquina para processar a tarefa j imediatamente após a tarefa i.

Todas as formulações anteriores continuam válidas ao se substituir (S3) por (S6) ou (S7). No caso de o tempo de preparação for dependente da sequência, o problema de minimização do *makespan*, representado por C_{max}, consiste na determinação do tempo total para processar todas as tarefas, isto é, $C_{max} = \max_{i \in N} C_i$ e pode ser formulado como:

$$\begin{array}{lllll} \min & C_{max} & & & \\ & C_{max} & \geq & C_i & \forall i \in N \\ & \text{(S1) - (S2)} & \text{- (S7)} & \text{- (S4) - (S5)} & \end{array}$$

2.24.8 Sequenciamento em uma máquina com penalidades por antecipação e atraso da produção

Apresentamos a seguir uma classe de problemas de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção (PSUMAA), o qual tem as seguintes características:

(a) Uma máquina deve processar um conjunto de n tarefas (*jobs*).

(b) Cada tarefa possui um tempo de processamento p_i, uma data inicial e_i e uma data final t_i desejadas para o término do processamento.

(c) A máquina executa no máximo uma tarefa por vez e uma vez iniciado o processamento de uma tarefa, a mesma deve ser finalizada, ou seja, não é permitido a interrupção de seu processamento.

(d) Todas as tarefas estão disponíveis para processamento na data 0.

(e) Quando uma tarefa j é sequenciada imediatamente após uma tarefa i, sendo estas pertencentes a diferentes famílias de produtos, é necessário um tempo s_{ij} para a preparação da máquina. Tempos de preparação de máquina nulos ($s_{ij} = 0$) implicam em produtos da mesma família. Assume-se, ainda, que a máquina não necessita de tempo de preparação inicial, ou seja, o tempo de preparação da máquina para o processamento da primeira tarefa na sequência é igual a 0.

(f) É permitido tempo ocioso entre a execução de duas tarefas consecutivas.

(g) As tarefas devem ser finalizadas dentro de uma janela de tempo $[e_i, t_i]$, denominada janela de entrega. Se a tarefa i for finalizada antes de e_i então há um custo de manutenção de estoque. Caso a tarefa seja finalizada após t_i então há associado um custo por atraso (que pode ser uma multa imposta por contratos de prestação de serviço), além de insatisfação do cliente. As tarefas que forem finalizadas dentro da janela de entrega não proporcionam nenhum custo para a empresa.

(h) Os custos por antecipação e atraso da produção são dependentes das tarefas, ou seja, cada tarefa possui um custo unitário de antecipação α_i e um custo unitário de atraso β_i.

(i) O objetivo a ser alcançado com a resolução deste problema é a minimização do somatório dos custos de antecipação e atraso da produção.

Para modelar este PSUMAA, sejam n o número de tarefas a serem processadas, p_i o tempo de processamento da tarefa i, I_i a data de início do processamento da tarefa i ($I_i \geq 0$) e s_{ij} o tempo de preparação da máquina necessário para processar a tarefa j imediatamente depois da tarefa i.

Sejam, ainda, 0 (zero) e $n+1$ duas tarefas fictícias, de tal forma que 0 antecede imediatamente a primeira tarefa e $n+1$ sucede imediatamente a última tarefa na sequência de produção. Admite-se que p_0 e p_{n+1} são iguais a zero e que $s_{0i} = 0$ e $s_{i,n+1} = 0, \quad \forall\, i = 1, \ldots, n$.

As restrições (2.22) garantem a existência de um tempo suficiente para completar uma tarefa i antes de começar uma tarefa j, caso uma tarefa j seja processada imediatamente após uma tarefa i, sem nenhuma tarefa intermediária.

$$I_j \geq I_i + y_{ij}(M + s_{ij}) + p_i - M \quad \forall\, i = 0, 1, \ldots, n \ , \ \forall\, j = 1, 2, \ldots, n+1 \ \text{ e } \ i \neq j \tag{2.22}$$

No conjunto de restrições (2.22), M é um valor adequadamente grande. A variável de decisão $y_{ij} \in \{0, 1\}$ é definida da seguinte forma:

$$y_{ij} = \begin{cases} 1, & \text{se a tarefa } j \text{ é sequenciada imediatamente após a tarefa } i; \\ 0, & \text{caso contrário.} \end{cases}$$

Assim, quando $y_{ij} = 1$, as restrições (2.22) tornam-se:

$$I_j \geq I_i + p_i + s_{ij} \tag{2.23}$$

No caso em que $y_{ij} = 0$, tem-se:

$$I_j \geq I_i + p_i - M \tag{2.24}$$

Nesta última situação, as restrições (2.22) ficam desativadas, pois a equação (2.24) é redundante, uma vez que a parcela $(I_j - I_i - p_i)$ será sempre maior que $-M$.

As restrições (2.25) e (2.26) garantem que cada tarefa tenha somente uma tarefa imediatamente antecessora e uma tarefa imediatamente sucessora, respectivamente.

$$\sum_{i=0,\, i\neq j}^{n} y_{ij} = 1 \quad \forall\, j = 1, 2, \ldots, n+1 \tag{2.25}$$

$$\sum_{j=1,\, j\neq i}^{n+1} y_{ij} = 1 \quad \forall\, i = 0, 1, \ldots, n \tag{2.26}$$

Sejam e_i a data de início do período de entrega da tarefa i, E_i o tempo de antecipação da tarefa i, t_i a data de término do período de entrega da tarefa i e T_i o tempo de atraso da tarefa i. As restrições (2.27) a (2.30) garantem que o tempo de antecipação E_i seja o máximo entre 0 e $e_i - p_i - I_i$ e que o tempo de atraso T_i seja o máximo entre 0 e $I_i + p_i - t_i$.

$$I_i + p_i + E_i \geq e_i \quad \forall\, i = 1, 2, \ldots, n \tag{2.27}$$

$$I_i + p_i - T_i \leq t_i \quad \forall\, i = 1, 2, \ldots, n \tag{2.28}$$

$$E_i \geq 0 \quad \forall\, i = 1, 2, \ldots, n \tag{2.29}$$

$$T_i \geq 0 \quad \forall\, i = 1, 2, \ldots, n \tag{2.30}$$

Sejam α_i e β_i os custos de antecipação e atraso da produção da tarefa i por unidade de tempo, respectivamente. O custo total por antecipação é dado por $\sum_{i=1}^{n} \alpha_i E_i$, enquanto o custo total por atraso é determinado por $\sum_{i=1}^{n} \beta_i T_i$. A função objetivo, que consiste em minimizar o somatório dos custos totais de antecipação e atraso da produção, é dado pela equação (2.31).

$$\min \quad z = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) \tag{2.31}$$

Resumindo, as variáveis de decisão do modelo relativo ao PSUMAA estudado são:

- I_i: data de início do processamento da tarefa i;
- y_{ij}: variável binária que determina a sequência de produção, se y_{ij}=1 a tarefa j é processada imediatamente depois da tarefa i e 0 caso contrário;
- E_i: tempo de antecipação da tarefa i;
- T_i: tempo de atraso da tarefa i;

Assim, o modelo correspondente de Programação Linear Inteira Mista (PLIM) para o PSU-MAA dado é:

$$\text{minimizar} \quad z = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) \tag{2.32}$$

$$\text{sujeito a:} \quad I_j - I_i - y_{ij}(M + s_{ij}) \geq p_i - M \quad \forall\, i = 0, 1, \ldots, n; \quad \forall\, j = 1, 2, \ldots, n+1 \; e \; i \neq j \tag{2.33}$$

$$\sum_{j=1,\, j\neq i}^{n+1} y_{ij} = 1 \qquad \forall\, i = 0, 1, \ldots, n \tag{2.34}$$

$$\sum_{i=0,\, i\neq j}^{n} y_{ij} = 1 \qquad \forall\, j = 1, 2, \ldots, n+1 \tag{2.35}$$

$$I_i + p_i + E_i \geq e_i \qquad \forall\, i = 1, 2, \ldots, n \tag{2.36}$$

$$I_i + p_i - T_i \leq t_i \qquad \forall\, i = 1, 2, \ldots, n \tag{2.37}$$

$$I_i \geq 0 \qquad \forall\, i = 0, 1, \ldots, n+1 \tag{2.38}$$

$$E_i \geq 0 \qquad \forall\, i = 1, 2, \ldots, n \tag{2.39}$$

$$T_i \geq 0 \qquad \forall\, i = 1, 2, \ldots, n \tag{2.40}$$

$$y_{ij} \in \{0, 1\} \qquad \forall\, i, j = 0, 1, 2, \ldots, n, n+1 \tag{2.41}$$

2.25 Máquinas Paralelas

Existem três categorias de ambientes de máquinas paralelas: idênticas, uniformes e não relacionadas. Em máquinas idênticas, todas as tarefas têm o mesmo tempo de processamento e exigem o mesmo tempo de preparação. Em máquinas uniformes, o tempo de processamento de uma tarefa e o tempo de preparação de máquinas mais modernas são proporcionais aos tempos correspondentes à máquina mais antiga. Em máquinas não relacionadas não existe uma relação entre os tempos de processamento e preparação de máquinas distintas.

Os modelos a seguir referem-se a m máquinas paralelas e n tarefas disponíveis para processamento no instante zero, sem interrupção de processamento de qualquer tarefa.

Para o problema de minimização do *makespan* em máquinas idênticas, seja p_i o tempo de processamento da tarefa i, $i = 1, \cdots, n$.

Sejam as seguintes variáveis de decisão:

C_{max} : *makespan*
x_{ik} : Variável binária que assume valor 1 se a tarefa i é processada na máquina k e 0, caso contrário

Assim, o modelo de programação matemática para a minimização do *makespan* em máquinas paralelas idênticas pode ser representado por:

$$\text{minimizar} \quad C_{max} \tag{2.42}$$

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad \forall\, i = 1 = 1, 2, \ldots, n \tag{2.43}$$

$$C_{max} \geq \sum_{i=1}^{n} p_i x_{ik} \quad \forall\, k = 1, 2, \cdots, m; \tag{2.44}$$

$$C_{max} \geq 0 \tag{2.45}$$

$$x_{ik} \in \{0, 1\} \qquad \forall\, i = 0, 1, 2, \ldots, n;\ k = 1, 2, \cdots, m \tag{2.46}$$

A função objetivo (2.42) representa a minimização do *makespan*. As restrições (2.43) asseguram que uma tarefa i é designada a exatamente uma máquina, enquanto as restrições (2.44) impõem que o *makespan* é o maior tempo de processamento entre todas as máquinas. As restrições (2.45) e (2.46) indicam o tipo das variáveis.

Apresenta-se, a seguir, um modelo para minimização da soma dos avanços e atrasos aplicável a qualquer dos três tipos de máquinas anteriormente mencionados. Para tanto, sejam os seguintes parâmetros de entrada:

p_{ik} : Tempo de processamento da tarefa i na máquina k
s_{ijk} : Tempo de preparação da máquina k para processar a tarefa j imediatamente após a tarefa i
d_i : Data de entrega da tarefa i
M : Número suficientemente grande

e as seguintes variáveis de decisão:

C_{ik} : Instante de término do processamento da tarefa i na máquina k
x_{ijk} : Variável binária que assume valor 1 se a tarefa i precede imediatamente a tarefa j na máquina k e 0, caso contrário
T_i : Atraso da tarefa i, isto é, $\max\{C_i - d_i, 0\}$
E_i : Avanço da tarefa i, isto é, $\max\{d_i - C_i, 0\}$

A variável T_i mensura o quanto a tarefa i está atrasada, enquanto E_i indica o quanto está adiantada em relação à data de entrega.

A seguir, o modelo de programação matemática para a minimização dos avanços e atrasos em máquinas paralelas.

$$\min \quad \sum_{i=1}^{n}(T_i + E_i) \tag{2.47}$$

$$\sum_{k=1}^{m}\sum_{i=0}^{n} x_{ijk} = 1 \qquad \forall\, j = 1, 2, \cdots, n \tag{2.48}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1 \qquad \forall\, k = 1, \cdots, m \tag{2.49}$$

$$\sum_{\substack{i=0 \\ i\neq h}}^{n} x_{ihk} - \sum_{\substack{j=0 \\ j\neq h}}^{n} x_{hjk} = 0 \qquad \forall\, h = 1, \cdots, n$$

$$\forall\, k = 1, 2, \cdots, m \tag{2.50}$$

$$C_{jk} \geq C_{ik} - M + (s_{ijk} + p_{jk} + M)x_{ijk} \quad \forall\, i = 0, 1, \cdots, n \tag{2.51}$$

$$\forall\, j = 1, 2, \cdots, n$$

$$\forall\, k = 1, 2, \cdots, m$$

$$T_i \geq C_i - d_i \qquad \forall\, i = 1, \cdots, n \tag{2.52}$$

$$E_i \geq d_i - C_i \qquad \forall\, i = 1, \cdots, n \tag{2.53}$$

$$T_i \geq 0 \qquad \forall\, i = 1, 2, \ldots, n \tag{2.54}$$

$$E_i \geq 0 \qquad \forall\, i = 1, 2, \ldots, n \tag{2.55}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall\, i, j = 0, 1, 2, \ldots, n \tag{2.56}$$

$$\forall\, k = 1, 2, \cdots, m$$

A função objetivo (2.47) expressa a minimização da soma total dos atrasos e avanços das tarefas. As restrições (2.48) impõem que cada tarefa j tem uma única tarefa predecessora imediata em uma única máquina. As restrições (2.49) garantem que cada máquina k, se usada, tem uma única sequência de processamento. As restrições (2.50) asseguram que cada tarefa j tem uma única tarefa sucessora imediata, com exceção da tarefa 0, que estabelece o início e o final da sequência de processamento na máquina k. Se $x_{ijk} = 1$, a restrição (2.51) implica que na máquina k tem-se $C_{jk} \geq C_{ik} + s_{ijk} + p_{jk}$ e, se $x_{ijk} = 0$, então $C_{jk} - C_{ik} \geq -M$, isto é, a restrição (2.51) fica desativada. As restrições (2.52) e (2.53) determinam as tarefas que estão com atrasos ou adiantamentos, respectivamente. As restrições (2.54), (2.55) e (2.56) indicam os tipos de variáveis.

2.26 *Job Shop*

Um *Job Shop* clássico é um ambiente de produção com n tarefas e m máquinas, em que cada tarefa é processada nas m máquinas, de acordo com um roteiro preestabelecido. Considere, por exemplo, 5 tarefas e 3 máquinas, denotadas por 1, 2 e 3. As matrizes O e P a seguir, representam, respectivamente, a matriz de operações, e a matriz de tempos de processamento nessas máquinas. Assim, por exemplo, a primeira linha da matriz O indica que a tarefa 1 é processada nas máquinas 2, 1 e 3, nesta ordem, com tempos de processamento de 5, 7 e 10 unidades de tempo, respectivamente, correspondentes aos elementos da primeira linha da matriz P.

$$O = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 2 \end{bmatrix} \qquad P = \begin{bmatrix} 5 & 7 & 10 \\ 9 & 5 & 3 \\ 5 & 8 & 2 \\ 2 & 7 & 4 \\ 8 & 8 & 8 \end{bmatrix}$$

Admita que as n tarefas estejam disponíveis para processamento no instante inicial e que não é permitida a interrupção do processamento de qualquer tarefa. Sejam os parâmetros:

p_{ik} : Tempo de processamento da tarefa i na máquina k
maq_i : máquina que processa a i-ésima tarefa
M : Número suficientemente grande

e as seguintes variáveis de decisão:

C_{ik} : Instante de término do processamento da tarefa i na máquina k
x_{ijk} : Variável binária que assume valor 1 se a tarefa i precede a tarefa j na máquina k e 0, caso contrário

Assim, o modelo de programação matemática relativo ao *Job Shop* pode ser representado por:

$$\text{minimizar} \quad \sum_{i=1}^{n} C_{im} \tag{2.57}$$

$$C_{i,maq_1} \geq p_{i,maq_1} \qquad \forall\, i = 1, 2, \ldots, n \tag{2.58}$$

$$C_{i,maq_{k+1}} \geq C_{i,maq_k} + p_{i,maq_{k+1}} \qquad \forall\, i = 1, 2, \ldots, n; \quad k = 1, \cdots, m-1 \tag{2.59}$$

$$C_{jk} \geq C_{ik} + p_{jk} - M(1 - x_{ijk}) \qquad \forall\, i, j = 1, 2, \cdots, n; \quad k = 1, 2, \cdots, m \tag{2.60}$$

$$C_{ik} \geq C_{jk} + p_{ik} - M x_{ijk} \qquad \forall\, i, j = 1, 2, \cdots, n; \quad k = 1, 2, \cdots, m \tag{2.61}$$

$$C_{ik} \geq 0 \qquad \forall\, i, j = 1, 2, \ldots, n; \quad k = 1, 2, \cdots, m \tag{2.62}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall\, i, j = 0, 1, 2, \ldots, n; \; k = 1, 2, \cdots, m \tag{2.63}$$

A função objetivo (2.57) expressa a minimização do tempo de fluxo total das tarefas. As restrições (2.58) garantem que a primeira operação de cada tarefa i é completada após o respectivo tempo de processamento. As restrições (2.59) impõem que a operação $k+1$ seja concluída depois do término da operação k e do tempo de processamento da operação $k+1$. As restrições (2.60) e (2.61) são restrições disjuntivas que indicam respectivamente se, na máquina k, a tarefa i precede a tarefa j, ou a tarefa j precede a tarefa i. Se $x_{ijk} = 1$ então, de (2.60) e (2.61), tem-se que: $C_{jk} \geq C_{ik} + p_{jk}$ e $C_{ik} - C_{jk} \geq p_{ik} - M$, isto é, o conjunto de restrições (2.61) é desativado. De modo análogo, se $x_{ijk} = 0$ então $C_{jk} - C_{ik} \geq p_{jk} - M$ e $C_{ik} \geq C_{jk} + p_{ik}$, isto é, as restrições (2.60) são desativadas. As restrições (2.62) e (2.63) estabelecem o tipo das variáveis.

2.27 Planejamento de lavra com Alocação Dinâmica de Caminhões

Nesse problema há um conjunto de frentes de minério e estéril, um conjunto de carregadeiras e um conjunto de caminhões. O objetivo é alocar as carregadeiras às frentes de lavra e determinar o número de viagens que cada caminhão deve fazer à cada frente de lavra de forma a atender a produção requerida de minério e estéril, bem como as características físicas e químicas desejadas para o produto a ser gerado. Normalmente, a produção (ou ritmo de lavra) é determinada para uma hora e replicada enquanto as condições da mina forem as mesmas. A cada carregadeira está associada uma produtividade máxima, em toneladas/hora e cada caminhão tem uma capacidade máxima de carregamento. São conhecidos os tempos de ciclo, em minutos, de cada caminhão à cada frente de lavra, bem como as especificações dos minérios, ditos parâmetros de controle, em

cada frente. Os caminhões só podem fazer viagens a frentes nos quais há carregadeiras compatíveis, porque existem carregadeiras que são pequenas e não têm altura suficiente para carregar determinados caminhões. A alocação dos caminhões é dita dinâmica no sentido de que ao descarregar, um caminhão pode se direcionar a uma frente de lavra diferente da viagem anterior. Este mecanismo de alocação permite aumentar a produtividade da frota de caminhões mas, em contrapartida, exige um sistema de despacho de caminhões. Considera-se, também, o atendimento a uma relação estéril/minério. Essa relação é requerida para a mina de modo a viabilizar a abertura de novas frentes e a realização de obras de infraestrutura.

Para a modelagem exata do problema, será utilizada a técnica de pesquisa operacional conhecida como programação por metas (*Goal Programming*). Para tanto, sejam os seguintes dados de entrada:

M : Conjunto das frentes de minério;
E : Conjunto das frentes de estéril;
F : Conjunto das frentes formado por $M \cup E$;
S : Conjunto dos parâmetros de controle analisados no minério;
$Carreg$: Conjunto dos equipamentos de carga (carregadeiras);
V : Conjunto dos equipamentos de transporte (caminhões);
Pr : Ritmo de lavra recomendado (t/h);
Pl : Ritmo de lavra mínimo (t/h);
Pu : Ritmo de lavra máximo (t/h);
β^- : Penalidade por desvio negativo da produção;
β^+ : Penalidade por desvio positivo da produção;
t_{ij} : Valor do parâmetro j na frente i (%);
tr_j : Teor recomendado para o parâmetro j na mistura (%);
tl_j : Teor mínimo admissível para o parâmetro j na mistura (%);
tu_j : Teor máximo admissível para o parâmetro j na mistura (%);
α_j^- : Penalidade por desvio negativo para o parâmetro j na mistura;
α_j^+ : Penalidade por desvio positivo para o parâmetro j na mistura;
rem : Relação estério/minério requerida;
Cu_k : Produção máxima do equipamento de carga k (t/h);
cap_l : Capacidade do caminhão l (t);
T_{il} : Tempo total de ciclo do caminhão l à frente i (min).
$txMax_l$ Taxa de utilização máxima permitida para o caminhão l (%);
$comp_{lk}$: $\begin{cases} 1 & \text{se o caminhão } l \text{ pode ser usado com o equipamento de carga } k; \\ 0 & \text{caso contrário.} \end{cases}$

e as seguintes variáveis de decisão:

x_i : Ritmo de lavra da frente i (t/h);
y_{ik} : $\begin{cases} 1 & \text{se o equipamento de carga } k \text{ for alocado à frente } i; \\ 0 & \text{caso contrário.} \end{cases}$
n_{il} : Número de viagens que o caminhão l realiza na frente i em uma hora;
dm_j^-: Desvio negativo de meta relativo ao parâmetro j na mistura (t/h);
dm_j^+: Desvio positivo de meta relativo ao parâmetro j na mistura (t/h);
P^- : Desvio negativo do ritmo de lavra em relação ao recomendado (t/h);
P^+ : Desvio positivo do ritmo de lavra em relação ao recomendado (t/h).

Desvio negativo (ou desvio por baixo) de meta do parâmetro de controle (respectivamente ritmo de lavra recomendado) indica o quanto se ficou abaixo da meta (respectivamente ritmo de

lavra recomendado), enquanto desvio positivo (ou desvio por cima) indica o quanto se ultrapassou a meta (respectivamente ritmo de lavra recomendado).

O modelo de programação matemática relativo à alocação dinâmica de uma frota heterogênea de caminhões e equipamentos de carga, levando-se em consideração metas de produção e qualidade de minério, é apresentado pelas equações (2.64)-(2.82).

$$\min \sum_{j \in S} \alpha_j^- dm_j^- + \sum_{j \in S} \alpha_j^+ dm_j^+ + \beta^- P^- + \beta^+ P^+ \tag{2.64}$$

s.a:

$$\sum_{i \in M} (t_{ij} - tu_j) x_i \leq 0 \qquad \forall j \in S \tag{2.65}$$

$$\sum_{i \in M} (t_{ij} - tl_j) x_i \geq 0 \qquad \forall j \in S \tag{2.66}$$

$$\sum_{i \in M} (t_{ij} - tr_j) x_i + dm_j^- - dm_j^+ = 0 \qquad \forall j \in S \tag{2.67}$$

$$\sum_{i \in M} x_i \leq Pu \tag{2.68}$$

$$\sum_{i \in M} x_i \geq Pl \tag{2.69}$$

$$\sum_{i \in M} x_i + P^- - P^+ = Pr \tag{2.70}$$

$$x_i \geq 0 \qquad \forall i \in F \tag{2.71}$$

$$dm_j^+, dm_j^- \geq 0 \qquad \forall j \in S \tag{2.72}$$

$$P^+, P^- \geq 0 \tag{2.73}$$

$$\sum_{i \in E} x_i - rem \sum_{i \in M} x_i \geq 0 \tag{2.74}$$

$$\sum_{k \in Carreg} y_{ik} \leq 1 \qquad \forall i \in F \tag{2.75}$$

$$\sum_{i \in F} y_{ik} \leq 1 \qquad \forall k \in Carreg \tag{2.76}$$

$$y_{ik} \in \{0, 1\} \qquad \forall i \in F, k \in Carreg \tag{2.77}$$

$$x_i - \sum_{k \in Carreg} Cu_k y_{ik} \leq 0 \qquad \forall i \in F \tag{2.78}$$

$$n_{il} T_{il} - 60 \sum_{k \in Carreg,\, comp_{lk} \neq 0} y_{ik} \leq 0 \qquad \forall i \in F,\ \forall l \in V \tag{2.79}$$

$$\sum_{i \in F} n_{il} T_{il} \leq 60 \times txMax_l \qquad \forall l \in V \tag{2.80}$$

$$x_i - \sum_{l \in V} n_{il} cap_l = 0 \qquad \forall i \in F \tag{2.81}$$

$$n_{il} \in \mathbb{Z}^+ \qquad \forall i \in F,\ \forall l \in V \tag{2.82}$$

Observa-se que (2.65)-(2.73) são restrições que juntamente com a função objetivo (2.64) formam o modelo de mistura de minérios com metas (vide exercício 2.19, página 35). A restrição (2.74) assegura o atendimento da relação estéril/minério mínima requerida. As demais restrições que complementam o modelo podem ser divididas em dois grupos. O primeiro diz respeito à alocação de equipamentos de carga e a faixa de produtividade que torna viável a utilização desses equipamentos. As restrições (2.75) definem que cada frente deve operar com no máximo um equipamento de carga, enquanto que as restrições (2.76) estabelecem que cada equipamento de carga deve operar em uma frente, no máximo. As restrições (2.77) fixam as variáveis y_{ik} como binárias. As restrições (2.78) limitam o ritmo de lavra máximo em cada frente em função da produtividade da carregadeira a ela alocada.

O segundo grupo de restrições está relacionado ao transporte de material na mina e à alocação dos caminhões. As restrições (2.79) fazem com que cada caminhão somente realize viagens à uma frente onde esteja alocado um equipamento de carga compatível. As restrições (2.80) definem que cada caminhão opere no máximo $txMax_l\%$ de sessenta minutos. As restrições (2.81) fazem com que o ritmo de lavra de uma frente seja igual à produção realizada pelos caminhões alocados a essa frente. As restrições (2.82) asseguram que o número de viagens que um caminhão faz a uma frente é um valor inteiro positivo.

2.28 Linearização do produto de variáveis binárias

Propriedade: O produto $z = x_1 x_2 \cdots x_n$ com $x_j \in \{0,1\} \quad \forall j = 1, 2, \cdots, n$ é equivalente a
$x_1 + x_2 + \cdots + x_n - z \leq n - 1, \quad z \leq x_j \ , \ x_j \in \{0,1\}, \forall j = 1, 2, \cdots, n, \quad z \geq 0.$

Exemplo: Aplicar esta propriedade ao problema não-linear 0-1 a seguir, transformando-o em um problema de programação linear inteira mista (PLIM).

$\min 3x_1 + 2x_2 + x_3 - 9x_1x_2 + 4x_1x_2x_3$

s.a: $x_j \in \{0,1\} \ \ \forall j = 1, 2, 3$

(a) Linearização do produto $z_1 = x_1x_2$:

Aplicando a propriedade ao produto das duas variáveis binárias, resulta:

$x_1 + x_2 - z_1 \leq 2 - 1 = 1$

$z_1 \leq x_1$

$z_1 \leq x_2$

$z_1 \geq 0$

(b) Linearização do produto $z_2 = x_1x_2x_3$:

De forma análoga, obtém-se:

$x_1 + x_2 + x_3 - z_2 \leq 3 - 1 = 2$

$z_2 \leq x_1$

$z_2 \leq x_2$

$z_2 \leq x_3$

$z_2 \geq 0$

Levando os resultados obtidos em (a) e (b) no modelo não-linear, obtém-se o seguinte problema de programação linear inteira mista equivalente:

$$
\begin{array}{llrlrlrlrlrl}
\min & & 3x_1 & + & 2x_2 & + & x_3 & - & 9z_1 & + & 4z_2 & \\
 & & x_1 & + & x_2 & & & - & z_1 & & & \leq \quad 1 \\
 & - & x_1 & & & & & + & z_1 & & & \leq \quad 0 \\
 & & & - & x_2 & & & + & z_1 & & & \leq \quad 0 \\
 & & x_1 & + & x_2 & + & x_3 & & & - & z_2 & \leq \quad 2 \\
 & - & x_1 & & & & & & & + & z_2 & \leq \quad 0 \\
 & & & - & x_2 & & & & & + & z_2 & \leq \quad 0 \\
 & & & & & - & x_3 & & & + & z_2 & \leq \quad 0 \\
 & & x_1 & , & x_2 & , & x_3 & & & & & \in \quad \{0,1\} \\
 & & & & & & & & z_1 & , & z_2 & \geq \quad 0
\end{array}
$$

3 *Branch-and-Bound*

O método *Branch-and-Bound* é um método de busca em árvore que se fundamenta na programação linear para explorar o espaço de busca. Em cada passso do método as variáveis inteiras são relaxadas e o subproblema resultante resolvido por um método da programação linear. Se a solução desse subproblema tiver todas as variáveis inteiras, então os descendentes do ramo da árvore analisado estão, naturalmente, implicitamente enumerados. Há outro critério para a *poda* da árvore também, mas trataremos disso mais adiante. Havendo variáveis não inteiras e falhando esse outro critério, é feita a escolha de uma variável a ramificar. Escolhida essa variável, deve-se, agora, escolher qual ramo da árvore explorar primeiro, aquele associado ao valor menor ou igual ao piso da variável (isto é, o ramo $x_j \leq \lfloor x_j \rfloor$) ou o ramo associado ao valor maior ou igual ao teto da variável (isto é, o ramo $x_j \geq \lfloor x_j \rfloor + 1$). Graficamente, isso significa dividir o espaço de soluções em dois subconjuntos tendo como elementos separadores o piso e o teto da variável ramificada. A região do espaço de busca entre o piso e o teto pode ser excluída, pois é desprovida de soluções inteiras.

Para ilustrar o método seja o seguinte problema de programação linear inteira.

$$\begin{array}{lccccc} \min & 4x_1 & + & 3x_2 & = & z \\ & 8x_1 & + & 3x_2 & \geq & 24 \\ & 5x_1 & + & 6x_2 & \geq & 30 \\ & x_1 & + & 2x_2 & \geq & 9 \\ & x_1 & , & x_2 & \in & \mathbb{Z}^+ \end{array}$$

A Figura 4 apresenta a árvore de *branching* desse problema de programação inteira, no qual se aplicou um critério, conhecido como Variante de Dank, para escolher a variável a ramificar. Este critério consiste em escolher para ramificar a variável não inteira que esteja mais próxima de um inteiro (tanto com relação ao piso quanto ao teto).

Considerando como regras busca em profundidade e analisar primeiramente o valor maior da variável escolhida para ser ramificada, isto é, o ramo $x_j \geq \lfloor x_j \rfloor + 1$, mostremos quais problemas de programação linear (PPLs) seriam necessários resolver aplicando-se a técnica *branch-and-bound*.

Como o PPL 1 não produziu solução com variáveis inteiras, então é escolhida uma variável para ramificar; que, no caso do Critério de Dank, é a variável x_2. Como $x_2 = 3,69$, então dois ramos têm que ser resolvidos: o ramo no qual $x_2 \leq 3$ e o ramo no qual $x_2 \geq 4$. Os valores de x_2 entre 3 e 4 são desprezados porque não conduzem a soluções inteiras. Pela regra estabelecida, devemos resolver primeiro o ramo da árvore associado a $x_2 \geq 4$.

O PPL 3 é formado, portanto, pelo PPL 1 relaxado incluída a restrição $x_2 \geq 4$. Resolvendo o PPL 3 encontramos uma solução que também não é inteira. Supondo busca em profundidade, deve-se, portanto, ramificar uma das variáveis não inteiras do PPL 3 (No caso de busca em largura, resolveríamos o PPL 2). No caso, apenas a variável x_1 é não inteira e dois ramos devem ser considerados: o associado a $x_1 \leq 1$ e o associado a $x_1 \geq 2$. Pela regra estabelecida, o ramo $x_1 \geq 2$ deve ser resolvido primeiro.

Resolvendo-se o PPL 4, o qual é formado pelo PPL 1 incluídas as restrições $x_2 \geq 4$ e $x_1 \geq 2$, obtém-se uma solução inteira com valor $z = 20$. Então esse ramo da árvore está definitivamente explorado, podendo-se fazer o *backtracking* e resolver o outro ramo imediato da árvore, descendente do PPL 3.

Resolvendo-se o PPL 5, o qual é descendente do PPL 3, encontra-se uma solução não inteira de valor $z = 20$. Ora, uma solução com esse valor já foi encontrada resolvendo-se o PPL 4. Resolvendo-o encontraríamos uma solução pior, então a árvore de busca pode ser *podada* no nó em questão. Como todos os ramos descendentes do PPL 1 associados à $x_2 \geq 4$ já foram analisados, ainda que implicitamente, resta analisar o ramo associado a $x_2 \leq 3$. Esse ramo deve ser analisado

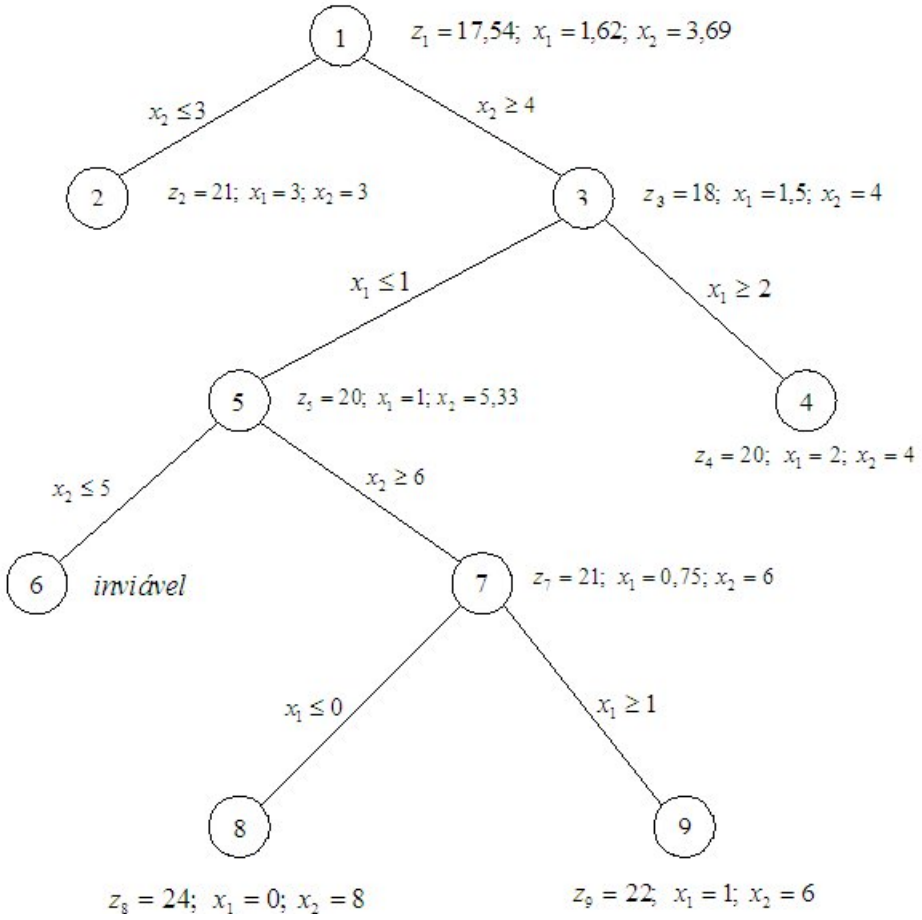


Figura 4: Árvore de *Branching*

porque sendo descendente do PPL 1, o qual tem valor $z = 17,54$, poderíamos encontrar valores de z iguais a 18 ou 19 e, portanto, inferiores ao melhor encontrado até o momento, que é $z = 20$. Entretanto, resolvendo o PPL 2 encontramos solução inteira, mas com $z = 21$. Nesse ponto, há a *poda* por dois motivos: primeiro, porque foi encontrada uma solução inteira e, segundo, porque o valor de z é pior que o melhor valor encontrado até o momento (Bastava a ocorrência de um desses motivos para que a *poda* pudesse ser realizada). Fazendo-se o *backtracking* retorna-se ao nó raiz, que é o PPL 1. Como os dois ramos da árvore foram analisados, o método é encerrado, retornando $x_1^\star = 2$, $x_2^\star = 4$, associado a $z^\star = 20$, como solução ótima para o problema de programação linear inteira dado.

Exercício:
Resolva o seguinte PLI pelo método *Branch-and-Bound*, apresentando toda a árvore de decisão. Use a variante de Dank para escolher a variável a ramificar, sendo que em caso de empate, escolha a variável de maior índice. Escolhida a variável a ramificar, opte por analisar primeiro o valor maior da variável. Faça busca em profundidade.

$$\begin{array}{lccccc} \max & 2x_1 & + & x_2 & = & z \\ & x_1 & + & 2x_2 & \le & 7 \\ & -x_1 & + & x_2 & \le & 0 \\ & 6x_1 & + & 2x_2 & \le & 21 \\ & x_1 & , & x_2 & \in & \mathbb{Z}^+ \end{array}$$

4 Integração do LINGO em planilhas Excel

Para ilustrar a integração do LINGO em planilhas do Microsoft Excel, consideremos o Problema de Transporte definido a seguir.

4.1 Problema de Transporte

Dado um conjunto de fontes de produção (fábricas), um conjunto de mercados consumidores (armazéns), e uma rede de possíveis caminhos de transporte (rotas) das fontes de produção para os mercados, o objetivo da problema é determinar o carregamento que minimiza o custo total de transporte, de modo que as capacidades das fontes produtivas não sejam ultrapassadas e as demandas dos mercados sejam atendidas. Considere a quantidade ofertada pelas fábricas maior que a soma das demandas dos armazéns.

Para fazer a modelagem de programação matemática deste problema, sejam os seguintes parâmetros de entrada:

$fabricas$	:	Conjunto das fábricas
$armazens$	:	Conjunto dos armazéns
$custo_{ij}$	:	Custo de transporte de uma unidade de produto da fábrica i ao armazém j
$demanda_j$	:	Demanda por produtos do armazém j
$capacidade_i$	:	Capacidade de produção da fábrica i

Considerando as variáveis de decisão $qtdEnviada_{ij}$ como sendo a quantidade de produtos a serem enviadas da fábrica i ao armazém j, temos o seguinte modelo de programação matemática:

$$\text{minimizar} \sum_{i \in fabricas} \sum_{j \in armazens} custo_{ij} \times qtdEnviada_{ij}$$

$$\sum_{j \in armazens} qtdEnviada_{ij} \leq capacidade_i \qquad \forall i \in fabricas$$

$$\sum_{i \in fabricas} qtdEnviada_{ij} = demanda_j \qquad \forall j \in armazens$$

$$qtdEnviada_{ij} \in Z^+ \qquad \forall i \in fabricas \; \forall j \in armazens$$

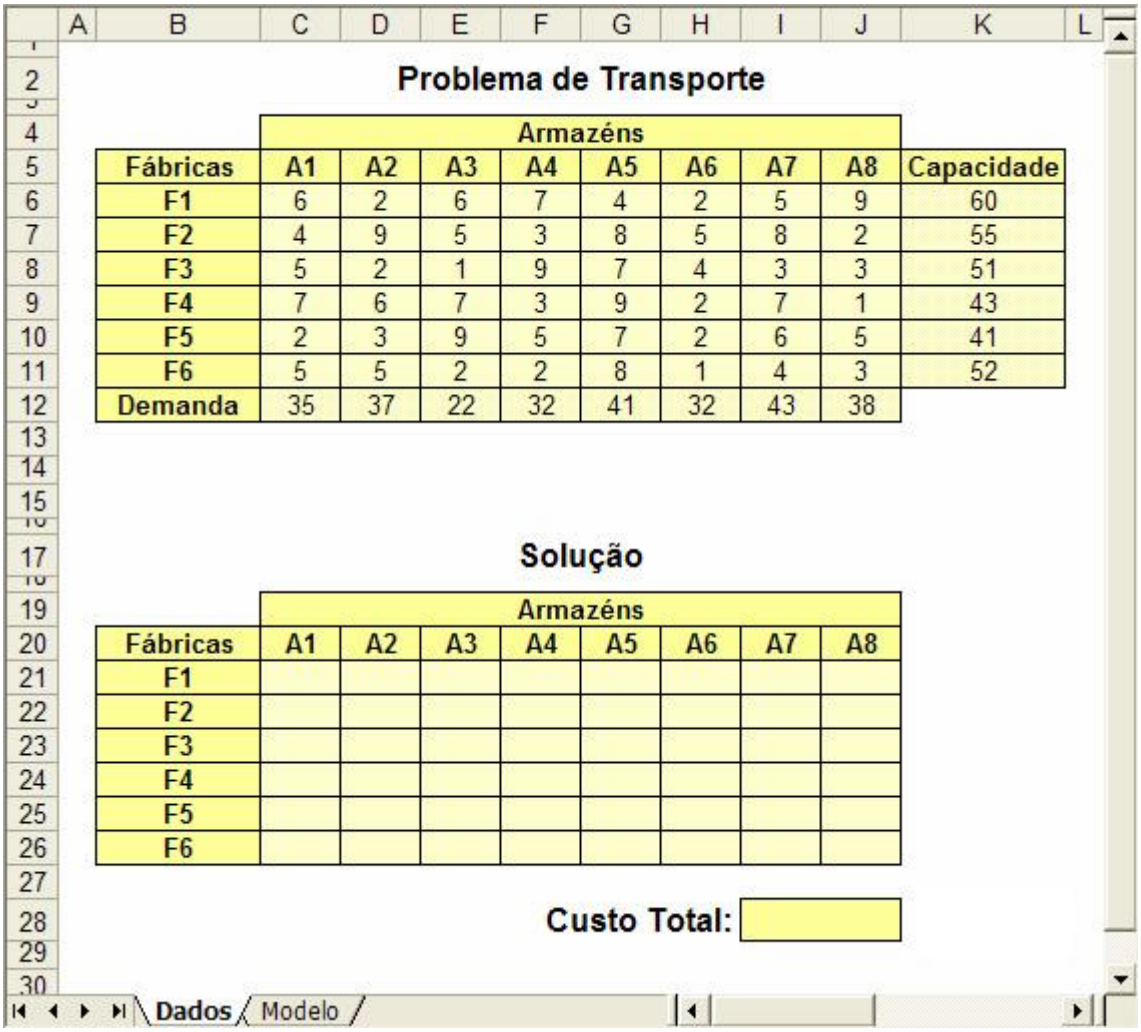
A última restrição, que estabelece o domínio das variáveis $qtdEnviada_{ij}$ no conjunto dos inteiros é desnecessária. O Problema de Transporte tem a propriedade de que qualquer solução do problema é inteira. Assim, esta última restrição pode ser substituída apenas por:
$qtdEnviada_{ij} \geq 0 \;\; \forall i \in fabricas \; \forall j \in armazens$

O modelo LINGO do Problema de Transporte é apresentado a seguir.

```
MODEL:
TITLE: Problema de Transporte;

SETS:
  fabricas / @OLE('Transporte.xlst','fabricas') /: capacidade;
  armazens / @OLE('Transporte.xls','armazens') / : demanda;
  rotas(fabricas, armazens): custo, qtdEnviada;
ENDSETS

DATA:
  capacidade = @OLE('Transporte.xls','capacidade');
  demanda = @OLE( 'Transporte.xls','demanda');
  custo = @OLE( 'Transporte.xls','custo');
ENDDATA
[FO] MIN = @SUM(rotas(i,j): custo(i,j)*qtdEnviada(i,j));

! As capacidades das fábricas não podem ser ultrapassadas;
@FOR(fabricas(i): @SUM(armazens(j): qtdEnviada(i,j)) <= capacidade(i));

! As demandas dos armazéns devem ser atendidas;
@FOR(armazens(j): @SUM(fabricas(i): qtdEnviada(i,j)) = demanda(j));

@FOR(rotas(i,j): @GIN(qtdEnviada(i,j));

DATA:
  @OLE('Transporte.xls','solucao') = qtdEnviada;
  @OLE( 'Transporte.xls','cTotal') = FO;
ENDDATA
END
```

Os dados de entrada para este modelo são apresentados pela figura a seguir. Nesta planilha estão definidos os seguintes campos:

Nome	**Campo**
fabricas	B6:B11
armazens	C5:J5
capacidade	K6:K11
demanda	C12:J12
custo	C6:J11
solucao	C21:J26
cTotal	I28



Problema de Transporte

	Armazéns								
Fábricas	A1	A2	A3	A4	A5	A6	A7	A8	Capacidade
F1	6	2	6	7	4	2	5	9	60
F2	4	9	5	3	8	5	8	2	55
F3	5	2	1	9	7	4	3	3	51
F4	7	6	7	3	9	2	7	1	43
F5	2	3	9	5	7	2	6	5	41
F6	5	5	2	2	8	1	4	3	52
Demanda	35	37	22	32	41	32	43	38	

Solução

	Armazéns							
Fábricas	A1	A2	A3	A4	A5	A6	A7	A8
F1								
F2								
F3								
F4								
F5								
F6								

Custo Total:

Dados | Modelo

4.2 Algumas considerações sobre @OLE

Ao utilizar a função @OLE, algumas considerações devem ser feitas:

1. Para que @OLE funcione corretamente é necessário que o arquivo ".xls" utilizado esteja aberto, a não ser quando objetos embutidos são utilizados (objetos embutidos são explicados na próxima seção);

2. A função @OLE não trabalha com conjuntos derivados tridimensionais; e

3. @OLE lê os campos definidos no Excel, de acordo com a seguinte ordem: da esquerda para direita e de cima para baixo.

4.3 Embutindo planilhas do EXCEL no LINGO

Assim como é possível embutir um modelo LINGO no EXCEL, o processo pode ser invertido de modo que uma planilha seja embutida no LINGO.

Para embutir um arquivo ".xls" no LINGO, siga os seguintes passos:

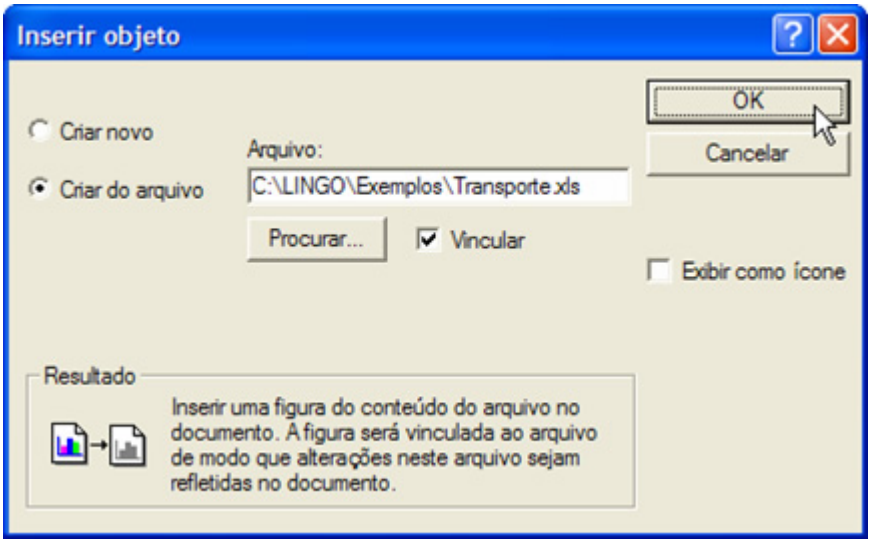
1. selecione o menu Edit|Insert New Object;

2. selecione a opção "Criar do Arquivo" na caixa de dialogo "Inserir Objeto";

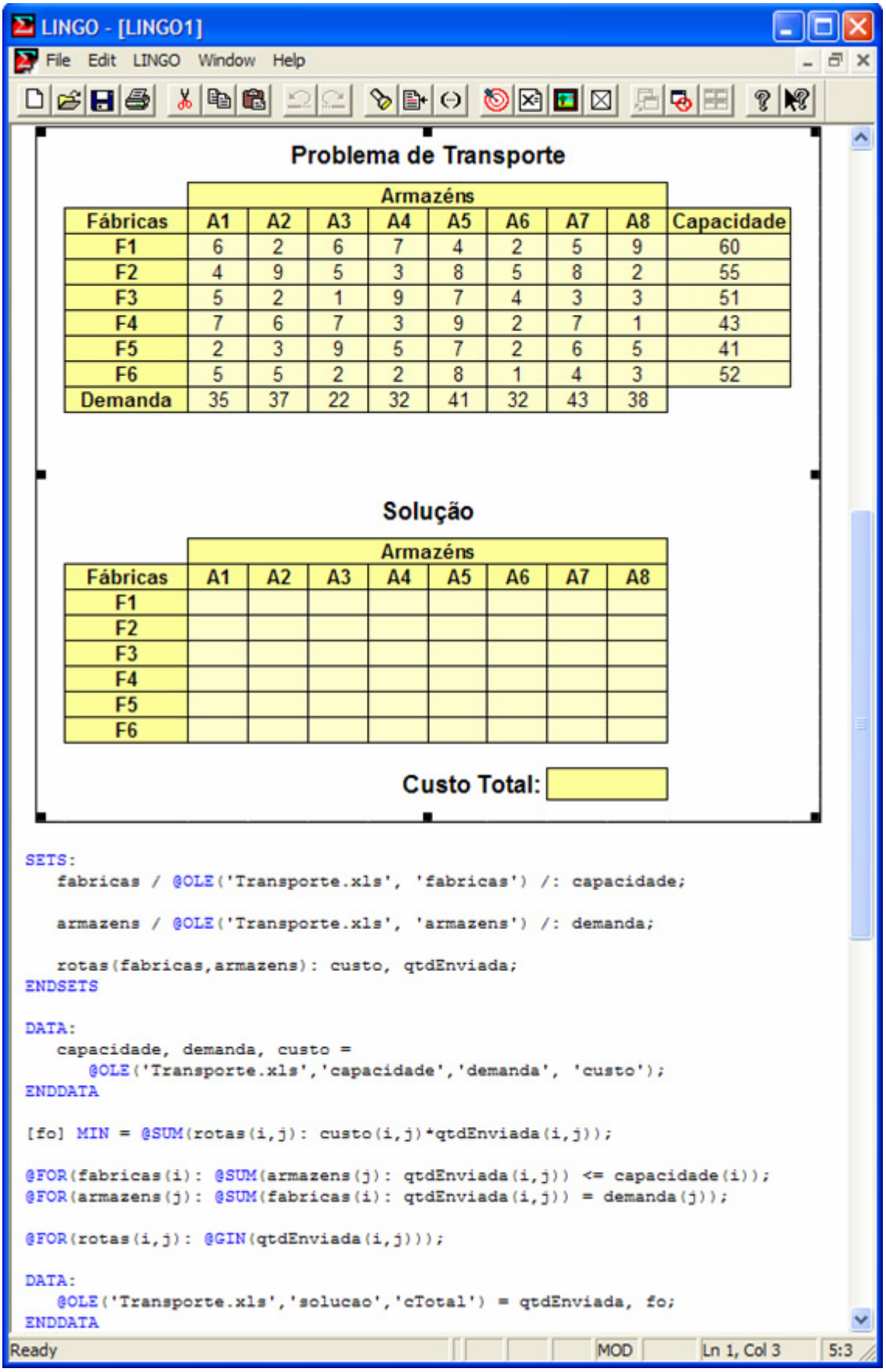
3. digite o caminho e o nome do arquivo a ser embutido;

4. marque a caixa "Vincular"; e

5. clique no botão OK.

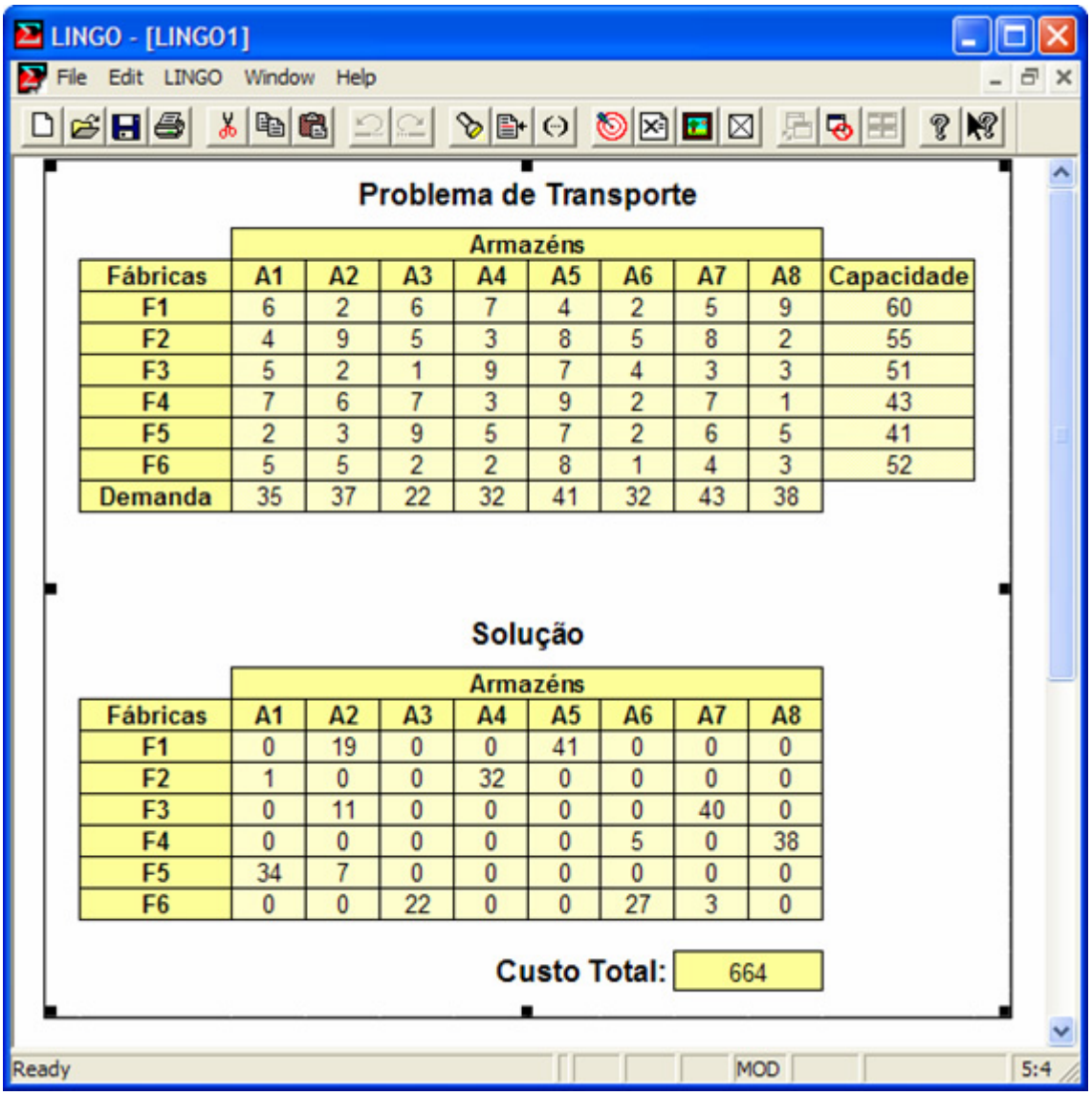


Este processo é ilustrado com o modelo "Problema de Transporte", apresentado na seção 4.1. Após inserir o novo objeto contendo o arquivo "Transporte.xls", temos:



A planilha de dados está agora embutida no LINGO, exibida ao topo do modelo "Problema de Transporte". Para editá-la, basta dar um duplo-clique sobre o objeto.

Quando o modelo for resolvido, o LINGO enviará os resultados para o arquivo "Transporte.xls" atualizando a planilha embutida, como exibido a seguir.

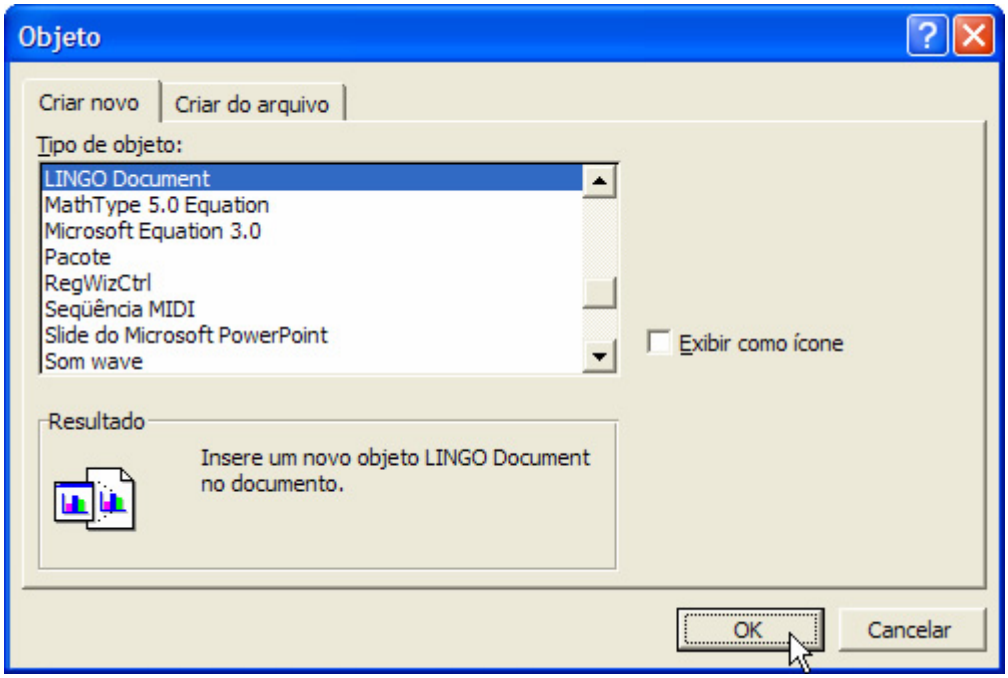


4.4 Embutindo Modelos LINGO no EXCEL

O LINGO é capaz de funcionar como um servidor OLE. Isto significa que um modelo do LINGO pode ser embutido em qualquer aplicação que funcione como cliente OLE como, por exemplo, o EXCEL. Embutir um modelo no EXCEL é conveniente, pois o modelo estará sempre disponível sempre que o arquivo ".xls" for aberto, não sendo necessário abrir o otimizador LINGO.

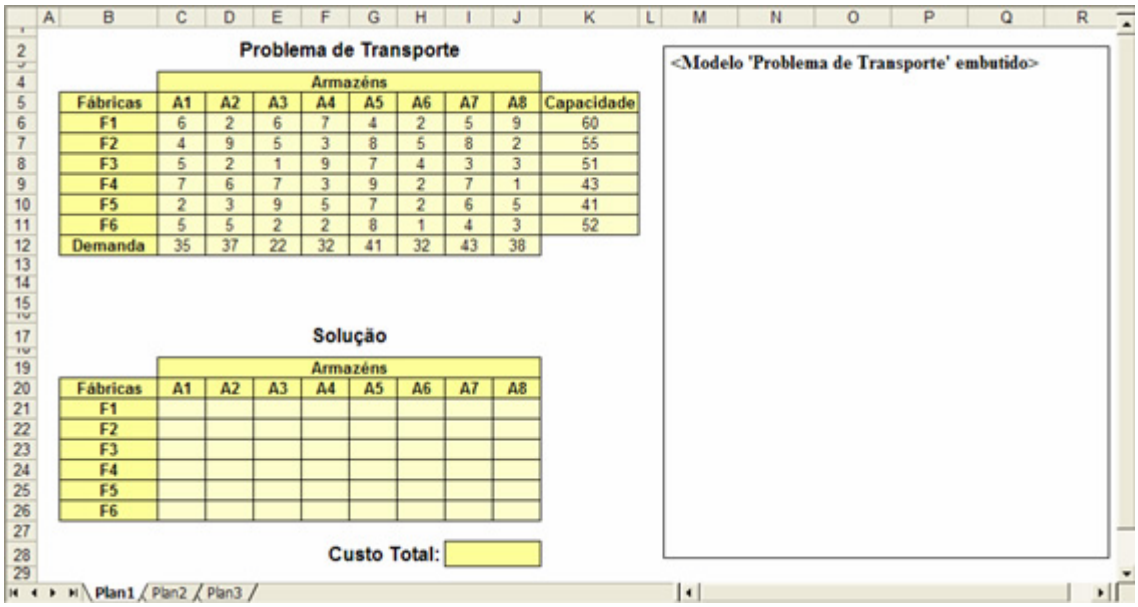
Para embutir um documento do LINGO em um arquivo do EXCEL, siga os seguintes passos:

1. selecione o menu Inserir|Objeto;
2. selecione o objeto "LINGO Document" na lista "Tipo de objeto"; e
3. clique no botão OK;

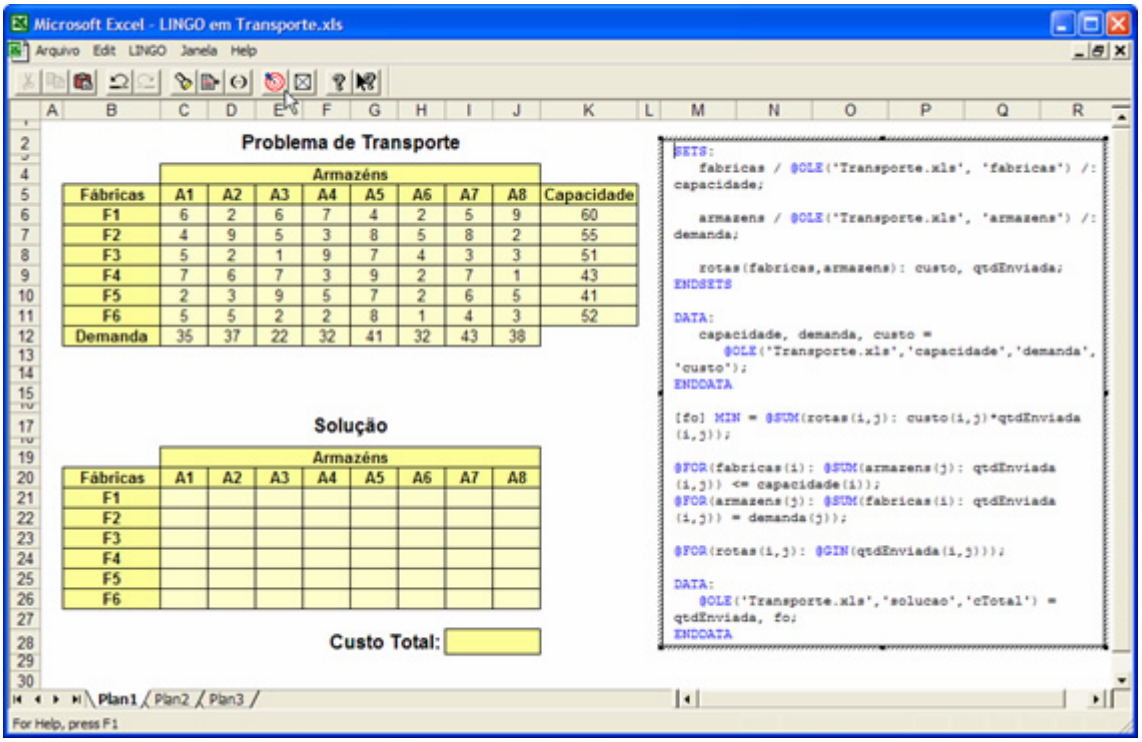


Após concluir os passos citados acima, um documento em branco do LINGO surgirá na planilha corrente. O modelo pode ser digitado no documento diretamente, ou copiado de uma outra aplicação (copiar/colar).

Para ilustrar este recurso, será utilizado o modelo "Problema de Transporte" descrito na seção 4.1. Embutindo este modelo em um arquivo nomeado como "Transporte.xls", teríamos:



Ao dar um duplo-clique sobre o objeto contendo o modelo embutido, uma barra de comandos do LINGO aparecerá no canto superior da tela. Para resolver o modelo, basta clicar no botão Solve da barra de comandos. Depois de otimizar o modelo, o LINGO enviará os resultados para o arquivo "Transporte.xls", como exibido a seguir.

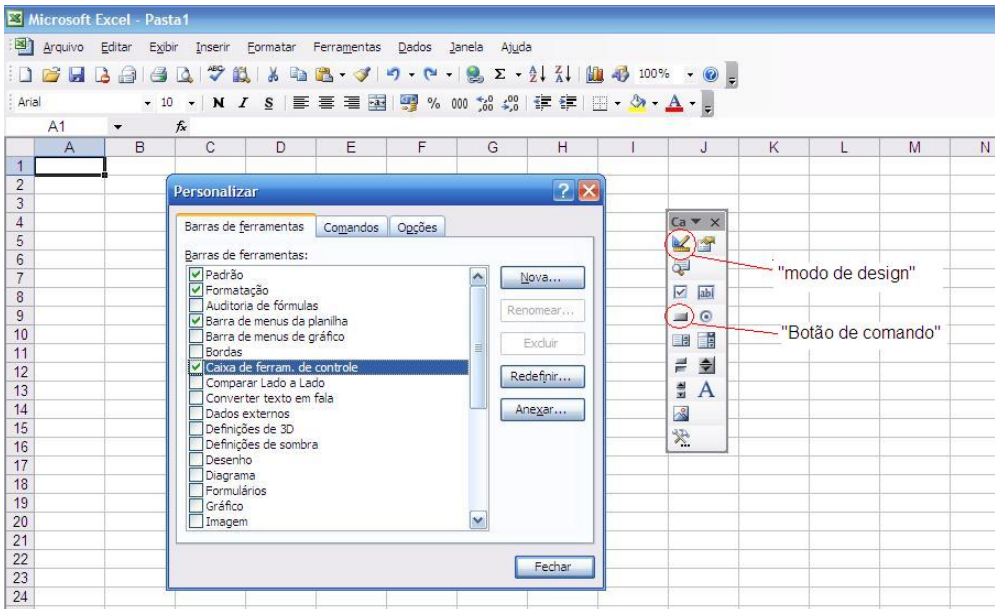


4.5 Utilizando links OLE automatizados no EXCEL

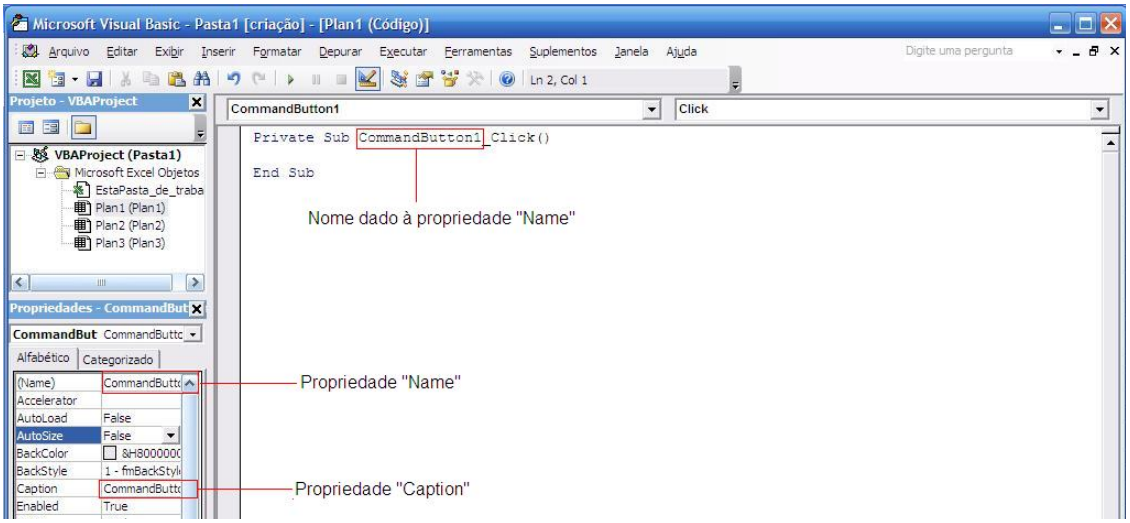
O LINGO disponibiliza um comando *script*, próprio para ser usado pelo EXCEL, que permite a criação de um *link* OLE automatizado. Este *link* estabelece uma relação cliente-servidor entre o EXCEL e o LINGO. Com isto, torna-se possível resolver um modelo escrito na própria planilha do EXCEL, sem a necessidade de utilizar o aplicativo do LINGO, de forma transparente para o usuário.

Para ilustrar esse recurso será utilizado o modelo "Problema de Transporte" mostrado na seção 4.1. Esta ilustração assume que o leitor esteja razoavelmente familiarizado com o uso de macros do Visual Basic.

Primeiramente, faça um modelo LINGO interfaceando com o Excel na forma usual. A seguir, vá na opção "Ferramentas / Personalizar" do Excel e ative o box "Caixa de Ferramentas de Controle".

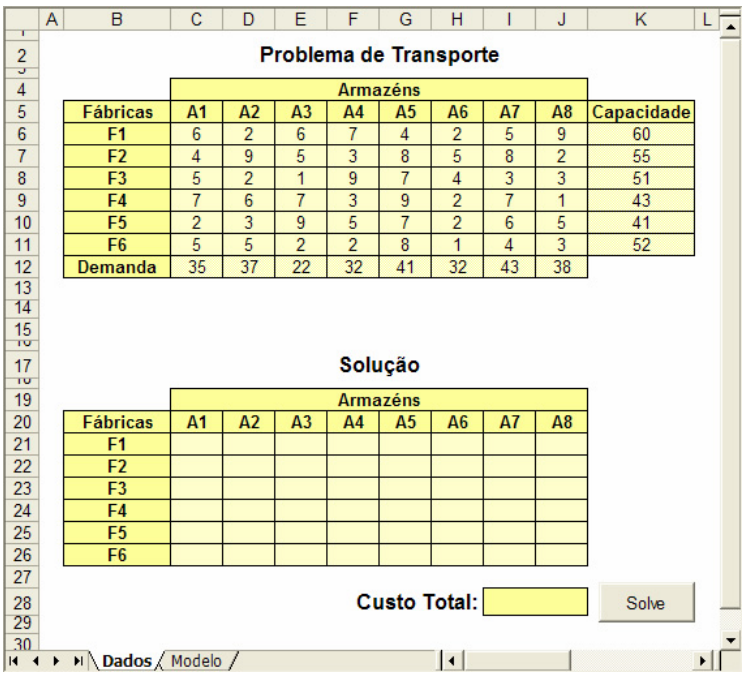


Ao aparecer a caixa de ferramenta, clique em **Botão de Comando** dessa Caixa e a seguir clique em uma posição adequada da planilha para inserir esse botão. Será, então, criado o botão **CommandButton1**. Dando um duplo clique nele, aparecerá a janela de edição do Visual Basic. Nesse editor, há uma janela à esquerda com as propriedades desse botão, conforme ilustra a figura a seguir.



Altere a propriedade **caption** para um nome adequado, por exemplo, **Solve** (observe que o nome anterior e *default* é "CommandButton1").

Na figura a seguir, representantiva da planilha Excel que faz interface com o Excel, aparece na coluna K28 o botão **Solve** resultante da alteração realizada.



Nesta planilha, intitulada "Dados", estão definidos os seguintes campos:

Nome	Campo	Nome	Campo
fabricas	B6:B11	custo	C6:J11
armazens	C5:J5	solucao	C21:J26
capacidade	K6:K11	cTotal	I28
demanda	C12:J12		

Existe ainda uma segunda planilha, chamada “Modelo”, que foi criada para conter o código *script* referente ao modelo do “Problema de Transporte”. Um *script* deve possuir o seguinte esquema:

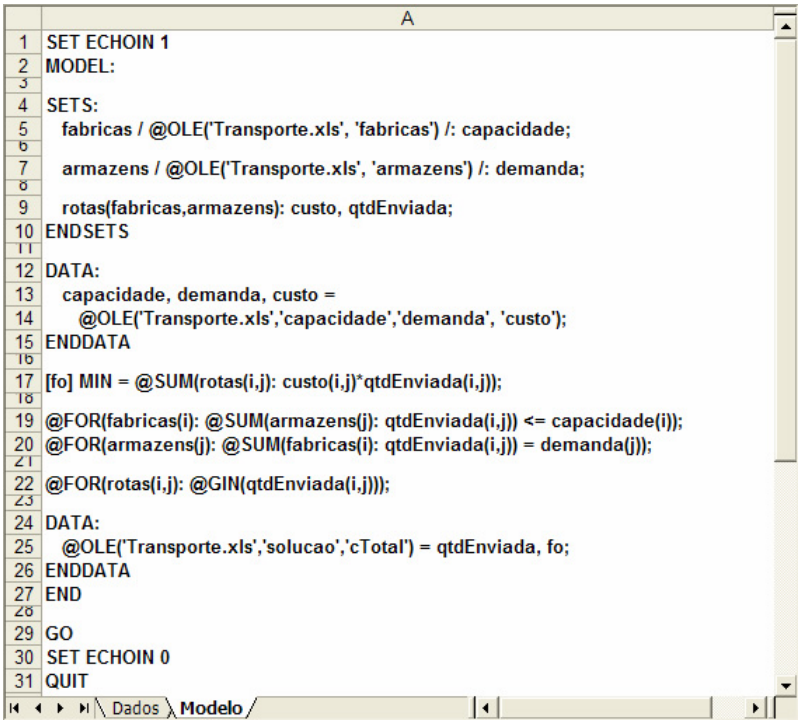
SET ECHOIN 1
Outras funções SET

MODEL:
modelo LINGO
END

GO
SET ECHOIN 0
QUIT

Observe que o modelo LINGO é delimitado pelas palavras-chave **MODEL** e **END**. O comando “SET ECHOIN 1” ativa o terminal do LINGO, permitindo que o *script* seja lido. Já o comando “GO” é usado para resolver o modelo, descrito entre as palavras-chave MODEL e END. O comando **QUIT** libera a memória usada pela aplicação LINGO.

Os comandos anteriores devem ser adicionados ao modelo LINGO e toda a relação de comandos, começando de SET ECHOIN 1 até QUIT deve ser marcada e copiada dentro do ambiente de modelagem LINGO com o comando **CRTL C**, por exemplo. Agora, posicione o cursor na planilha “Modelo” do Excel em uma certa posição, no caso, na célula A1. Dê **CRTL V**. A figura a seguir exibe o resultado dessa operação na planilha “Modelo”:



Para que este *script* seja enviado ao LINGO é necessário que ele esteja definido através do seguinte campo:

Nome	**Campo**
modelo	A1:A31

Isto é, o bloco do Excel onde está o código LINGO deve ser marcado e nomeado como "modelo" ou outro nome qualquer.

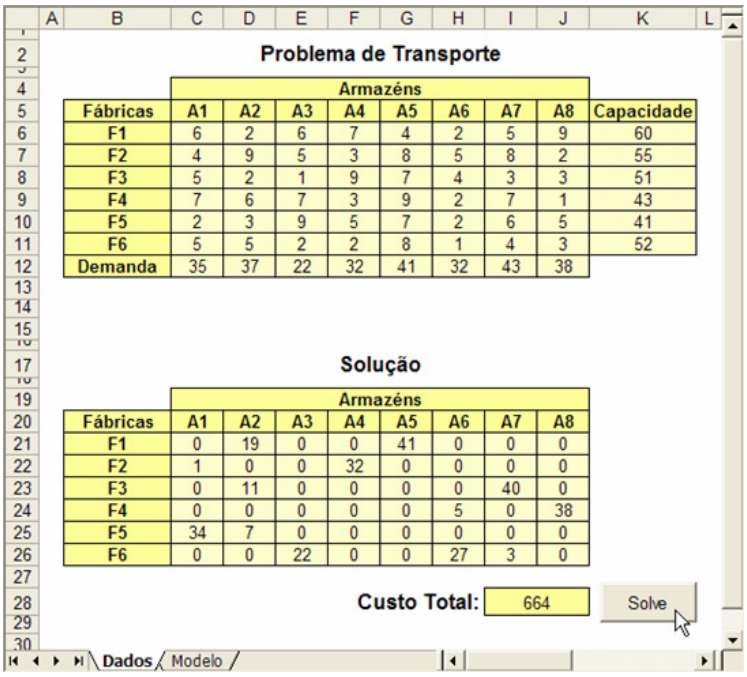
Definidos os campos e o modelo LINGO, será necessário associar ao botão **Solve**, criado na planilha "Dados", o seguinte código:

```
Private Sub CommandButton1_Click()
  Dim iErr As Integer
  Dim LINGO As Object

  Set LINGO = CreateObject("LINGO.Document.4")
  iErr = LINGO.RunScriptRange("modelo")

  If (iErr > 0) Then
    MsgBox ("Erro. O modelo não pode ser Resolvido")
  End If
End Sub
```

Se o editor Visual Basic não estiver aberto, este pode ser acionado dando-se um duplo clique no botão **Solve**. Os códigos anteriores supõem que a propriedade **Name** é "CommandButton1". Caso ela seja mudada para "Solve", então a primeira linha deverá ser **Private Sub Solve_Click()**.

A automação OLE é utilizada para chamar o método "RunScriptRange", passando o campo **modelo** como parâmetro. A rotina "RunScriptRange" então, solicita ao EXCEL que obtenha o conteúdo deste campo e, inicia o processo de execução do *script*. Esse processo continua até que a palavra-chave "QUIT" seja encontrada ou não haja mais comando a ser lido. A instrução "RunScriptRange" retornará um valor 0 caso o *script* esteja pronto para ser processado.

Voltando à planilha "Dados", para que o modelo seja resolvido basta apenas que o botão Solve seja pressionado. Após uma breve pausa, a solução encontrada pelo LINGO é enviada à planilha, como mostra a figura a seguir.



Problema de Transporte

	Armazéns								
Fábricas	A1	A2	A3	A4	A5	A6	A7	A8	Capacidade
F1	6	2	6	7	4	2	5	9	60
F2	4	9	5	3	8	5	8	2	55
F3	5	2	1	9	7	4	3	3	51
F4	7	6	7	3	9	2	7	1	43
F5	2	3	9	5	7	2	6	5	41
F6	5	5	2	2	8	1	4	3	52
Demanda	35	37	22	32	41	32	43	38	

Solução

	Armazéns							
Fábricas	A1	A2	A3	A4	A5	A6	A7	A8
F1	0	19	0	0	41	0	0	0
F2	1	0	0	32	0	0	0	0
F3	0	11	0	0	0	0	40	0
F4	0	0	0	0	0	5	0	38
F5	34	7	0	0	0	0	0	0
F6	0	0	22	0	0	27	3	0

Custo Total: 664 Solve

Dados Modelo

Feche o Editor do Visual Basic e retorne à planilha Excel. Para sair do modo de desenvolvimento, clique no botão **Sair do modo de design** do box "Caixa de Ferramentas de Controle".

Pronto, o botão **Solve** está preparado para uso. Feche o LINGO, pois não é preciso que ele esteja aberto para o botão funcionar. A seguir clique no botão **Solve** e a resposta será enviada para a planilha automaticamente.

4.6 Comando SET

O comando SET permite alterar configurações padrões do LINGO. Todas as opções configuráveis pelo usuário estão disponíveis através desse comando. Sua sintaxe é:

SET nome_do_parametro | índice_do_parametro [valor_do_parametro]

Caso o valor do parâmetro seja omitido, o LINGO utilizará o valor padrão para o parâmetro especificado. Alguns dos parâmetros acessíveis através do comando SET são apresentados a seguir.

Índice	**Nome**	**Padrão**	**Descrição**
10	TIMLIM	0	Tempo limite de execução em segundos (0: sem limite)
23	TERSEO	0	Omite o relatório gerado após a resolução do modelo (0: não, 1: sim)
24	STAWIN	1	Exibe a janela de *status* do processo de busca (1: sim, 0: não)
33	ECHOIN	0	Envia comandos *script* para o terminal (0: não, 1: sim)
34	ERRDLG	1	Exibe mensagens de erro em uma caixa de diálogo (1: sim, 0: não)
46	DUALCO	1	Calcula os valores duais (0: não calcula, 1: calcula só dual, 2: calcula dual e "range")
51	CUTOFF	$1 \times 10^{-}9$	Qualquer solução com valor menor ou igual a CUTOFF é considerada como zero
41	SOLVEL	0	Escolhe o resolvedor dos PPL's (0: LINGO decide, 1: Primal Simplex 2: Dual Simplex, 3: Barreira)
40	PRBLVL	0	Em programação inteira mista, realiza a operação *probing*, isto é, tenta deduzir um valor mais próximo de um inteiro para acelerar a busca. Pode surtir o efeito contrário. (0: LINGO escolhe, 1: nível mais baixo, 7: nível mais alto)
18	IPTOLR	5×10^{-8}	Esta tolerância é um valor r variando entre 0 e 1, que indica ao método B & B para somente buscar soluções inteiras cujo valor seja pelo menos $100 \times r\%$ melhor que a melhor solução inteira encontrada até então. Acelera a busca, mas pode não garantir a solução ótima.
17	IPTOLA	8×10^{-8}	Esta tolerância é um valor r, que indica ao método B & B para somente buscar soluções inteiras cujo valor seja pelo menos r unidades melhores que a melhor solução inteira encontrada até então. Acelera a busca, mas pode não garantir a solução ótima.
16	HURDLE	*none*	Valor de uma solução, normalmente encontrado via uma heurística. Com esse valor, o método B & B não explora soluções piores que ele. Assim, serve para acelerar a busca.

5 Problema do Caixeiro Viajante

5.1 Definição

Dado um conjunto de n cidades e uma matriz de distâncias d_{ij} entre elas, o Problema do Caixeiro Viajante (PCV), ou *Traveling Salesman Problem - TSP*, consiste em estabelecer uma rota para um Caixeiro, iniciando seu percurso em uma cidade, chamada cidade origem, passar por todas as demais $n-1$ cidades uma única vez e retornar à cidade origem percorrendo a menor distância possível.

Exemplificando, consideremos 6 cidades com as distâncias dadas pela tabela a seguir:

Cidade	1	2	3	4	5	6
1	0	2	1	4	9	1
2	2	0	5	9	7	2
3	1	5	0	3	8	6
4	4	9	3	0	2	5
5	9	7	8	2	0	2
6	1	2	6	5	2	0

Uma possível solução para o exemplo considerado é $s = (1\ \ 4\ \ 2\ \ 5\ \ 3\ \ 6)$. Para esta solução, a distância total percorrida é $dist = d_{14} + d_{42} + d_{25} + d_{53} + d_{36} + d_{61} = 4 + 9 + 7 + 8 + 6 + 1 = 35$. Observe que qualquer permutação das n cidades representa uma solução para o PCV. O que queremos é, dentre todas as possíveis permutações (soluções), determinar aquela cuja distância total percorrida é a menor possível.

Se $d_{ij} = d_{ji}$ diz-se que o PCV é simétrico. Caso contrário ele é dito assimétrico.

Para o PCV simétrico há $(n-1)!/2$ soluções possíveis. Para mostrar a magnitude do espaço de soluções, para $n = 20$ há 6×10^{16} soluções. Supondo que um computador avalie uma solução (rota) em 10^{-8} segundos, seriam necessários 6×10^{8} segundos ou 168951 horas ou 7039 dias ou cerca de 19 anos para se encontrar a melhor solução por enumeração completa de todas as possíveis soluções.

5.2 Modelagem de Programação Matemática

Seja o grafo $G = (Cidades, A)$, onde $Cidades$ é conjunto de cidades (clientes) e A o conjunto de arcos ligando duas cidades, isto é, $A = \{(i,j) \mid i \neq j\}$.

Seja d_{ij} a distância da cidade i para a cidade j.

(a) Variáveis de decisão:

x_{ij}: variável binária que assume valor 1 se o arco (i,j) for utilizado e 0, caso contrário

f_{ij}: quantidade de fluxo enviada da cidade i para a cidade j

(b) Função objetivo:

$$\min \sum_{i \in Cidades} \sum_{j \in Cidades} d_{ij} x_{ij}$$

(c) Restrições:

c.1) À cada cidade k só chega um arco:

$$\sum_{i \in Cidades} x_{ik} = 1 \ \ \forall k \in Cidades$$

c.2) De cada cidade k só sai um arco:
$$\sum_{j \in Cidades} x_{kj} = 1 \ \ \forall k \in Cidades$$

c.3) Eliminação de subciclos:
$$\sum_{i \in Cidades} f_{ik} - \sum_{j \in Cidades} f_{kj} = 1 \ \ \forall k \in Cidades \mid k \neq 1$$

$$f_{ij} \leq (n-1)x_{ij} \ \ \forall i \in Cidades, \ \ \forall j \in Cidades$$

c.4) Integralidade e não-negatividade:
$x_{ij} \in \{0,1\} \ \ \forall i \in Cidades, \ \ \forall j \in Cidades$
$f_{ij} \geq 0 \ \ \forall i \in Cidades, \ \ \forall j \in Cidades$

Observe que nas restrições de eliminação de subciclos, a variável contínua f indica o fluxo em um arco. Na primeira delas, impõe-se que o fluxo que chega a uma cidade k menos o que sai de k seja igual a 1 (exceto para a cidade origem, cujo índice é 1). Já na segunda, o fluxo máximo que passa em um arco usado no percurso é inferior a $n-1$, onde n é o número de cidades, e quando um arco não é usado ($x_{ij} = 0$) então o fluxo é nulo.

O modelo LINGO completo do PCV é apresentado a seguir. Neste modelo, V é o conjunto de cidades e PCV.txt (vide Figura 5) é um arquivo texto contendo os dados do PCV, a saber: relação das cidades (A, $\cdots$, P) e suas coordenadas cartesianas.

```
! Cidades;
A B C D E F G H I J K L M N O P ~

! Coordenadas x;
30 37 49 52 20 40 21 17 31 52 51 42 31  5 12 36 ~

! Coordenadas y;
40 52 49 64 26 30 47 63 62 33 21 41 32 25 42 16
```

Figura 5: Arquivo PCV.txt

```
model:
title: Problema do Caixeiro Viajante;
sets:
  V / @file('Coordenadas.txt') /: u, coord_x, coord_y;
  Matriz(V, V): d, x;
endsets

data:
  coord_x = @file('PCV.txt');
  coord_y = @file('PCV.txt');
enddata

@for(V(i):
   x(i,i) = 0;
   @for(V(j):
      d(i,j) = ( (coord_x(j) - coord_x(i))^2 +
                 (coord_y(j) - coord_y(i))^2   )^(0.5)));

[fo] min = @sum(V(i): @sum(V(j): d(i,j) * x(i,j)));

! A cada cidade k só chega um arco;
@for(V(k):
   @sum(V(i): x(i,k)) = 1);

! De cada cidade k só sai um arco;
@for(V(k):
   @sum(V(j): x(k,j)) = 1);


! Restricoes de eliminacao de subciclos;
@for(V(k) | k #ne# 1:
    @sum(V(i): f(i,k)) - @sum(V(j): f(k,j)) = 1);

@for(V(i):
  @for(V(j): f(i,j) <= (@size(V) - 1) * x(i,j)));

@for(V(i):
   @for(V(j): @bin(x(i,j))));

end
```

A solução ótima deste problema tem distância total fo $= 219,45$, sendo a rota ótima dada por A $\rightarrow$ M $\rightarrow$ F $\rightarrow$ J $\rightarrow$ K $\rightarrow$ P $\rightarrow$ E $\rightarrow$ N $\rightarrow$ O $\rightarrow$ G $\rightarrow$ H $\rightarrow$ I $\rightarrow$ B $\rightarrow$ D $\rightarrow$ C $\rightarrow$ L $\rightarrow$ A.

5.3 Modelagem Heurística

5.3.1 Heurísticas Construtivas

Uma heurística construtiva tem por objetivo construir uma solução, elemento por elemento. A forma de escolha de cada elemento a ser inserido a cada passo varia de acordo com a função de avaliação adotada, a qual, por sua vez, depende do problema abordado. Nas heurísticas clássicas, os elementos candidatos são geralmente ordenados segundo uma função gulosa, que estima o benefício da inserção de cada elemento, e somente o "melhor" elemento é inserido a cada passo.

É importante mencionar que não há garantia de que a solução final produzida por uma heurística seja a ótima.

A figura a seguir mostra o pseudocódigo para a construção de uma solução inicial para um problema de otimização que utiliza uma função de avaliação $g(.)$. Nesta figura, t_{melhor} indica o membro do conjunto de elementos candidatos com o valor mais favorável da função de avaliação g, isto é, aquele que possui o menor valor de g no caso de o problema ser de minimização ou o maior valor de g no caso de o problema ser de maximização.

```
procedimento ConstrucaoGulosa(g(.), s);
1  s ← ∅;
2  Inicialize o conjunto C de elementos candidatos;
3  enquanto (C ≠ ∅) faça
4      g(t_melhor) = melhor{g(t) | t ∈ C};
5      s ← s ∪ {t_melhor};
6      Atualize o conjunto C de elementos candidatos;
7  fim-enquanto;
8  Retorne s;
fim ConstrucaoGulosa;
```

Ilustraremos o PCV com três heurísticas construtivas, a saber: (a) Heurística do Vizinho Mais Próximo; (b) Heurística de Nemhauser e Bellmore e (c) Heurística da Inserção Mais Barata.

(a) **Heurística do Vizinho Mais Próximo**:
Nesta heurística, parte-se da cidade origem e adiciona-se a cada passo a cidade k ainda não visitada cuja distância à última cidade visitada é a menor possível. O procedimento de construção termina quando todas as cidades forem visitadas, situação na qual é feita a ligação entre a última cidade visitada e a cidade origem.

No exemplo considerado, considerando-se a cidade 1 como a cidade origem, tem-se:

i) Passo 1: Adicione a cidade 3 à rota, já que sua distância à cidade 1 é a menor (A cidade 6 também tem mesma distância, e também poderia ser escolhida).

ii) Passo 2: Adicione a cidade 4 à rota, já que sua distância à cidade 3 é a menor dentre as cidades ainda não visitadas (no caso, as cidades 2, 4, 5 e 6).

iii) Passo 3: Adicione a cidade 5 à rota, já que sua distância à cidade 4 é a menor dentre todas as cidades ainda não visitadas (no caso, as cidades 2, 5 e 6)

iv) Passo 4: Adicione a cidade 6 à rota, já que sua distância à cidade 5 é a menor dentre todas as cidades ainda não visitadas (no caso, as cidades 2 e 6)

v) Passo 5: Adicione a cidade 2 à rota, já que esta é a única cidade ainda não visitada

vi) Passo 6: Faça a ligação da cidade 2 (última cidade visitada) à cidade 1 (cidade origem)

Ao final desses 6 passos, teremos produzido a solução $s = (1\ \ 3\ \ 4\ \ 5\ \ 6\ \ 2)$. Para esta solução, a distância total percorrida é $dist = d_{13} + d_{34} + d_{45} + d_{56} + d_{62} + d_{21} = 1+3+2+2+2+2 = 12$.

Complexidade da Heurística do Vizinho Mais Próximo:

Iteração	# operações	Observações
1	$n-1$	Há $n-1$ ligações para serem analisadas
2	$n-2$	Há $n-2$ ligações para serem analisadas
$\cdots$	$\cdots$	$\cdots$
$n-1$	1	Há apenas uma cidade ainda não visitada
Total	$1 + 2 + \cdots + n - 1$	$= n(n-1)/2$ operações

A soma anterior é uma Progressão Aritmética cujo primeiro elemento é 1, último elemento é $n-1$, a razão é igual a 1 e o número de termos é $n-1$. A soma dos termos desta PA vale $S = \left(\frac{a_1 + a_{nelem}}{2}\right) nelem = \left(\frac{1+(n-1)}{2}\right)(n-1) = n(n-1)/2$

(b) **Heurística de Bellmore e Nemhauser**:
Nesta heurística, adicionamos à rota corrente a cidade k ainda não visitada que esteja mais próxima dos extremos da subrota, isto é, a cidade k se liga a uma cidade que esteja em uma extremidade da subrota ou à outra extremidade.

No exemplo considerado, considerando-se a cidade 1 como a cidade origem, tem-se:

i) Passo 1: Adicione a cidade 3 à rota, já que sua distância à cidade 1 é a menor (A cidade 6 também tem mesma distância, e também poderia ser escolhida).

ii) Passo 2: Das cidades ainda não visitadas (2, 4, 5 e 6), a cidade 6 é a que menos dista de um extremo da rota (cidade 1) e a cidade 4 é a que menos dista do outro extremo da rota (cidade 3). Como a distância $d_{61} = 1 < d_{34} = 3$, então a cidade 6 é a escolhida e deve ser conectada à cidade 1, isto é, a rota corrente é: $s = (6 \to 1 \to 3)$.

iii) Passo 3: Das cidades ainda não visitadas (2, 4 e 5), a cidade 2 é a que menos dista de um extremo da rota (cidade 6) e a cidade 4 é a que menos dista do outro extremo da rota (cidade 3). Como a distância $d_{26} = 2 < d_{34} = 3$, então a cidade 2 é a escolhida e deve ser conectada à cidade 6, isto é, a rota corrente é: $s = (2 \to 6 \to 1 \to 3)$. A cidade 5 também poderia ter sido escolhida para se conectar à cidade 6, pois tem a mesma distância da cidade 2 à cidade 6.

iv) Passo 4: Das cidades ainda não visitadas (4 e 5), a cidade 5 é a que menos dista de um extremo da rota (cidade 2) e a cidade 4 é a que menos dista do outro extremo da rota (cidade 3). Como a distância $d_{34} = 3 < d_{52} = 7$, então a cidade 4 é a escolhida e deve ser conectada à cidade 3, isto é, a rota corrente é: $s = (2 \to 6 \to 1 \to 3 \to 4)$.

v) Passo 5: A única cidade ainda não visitada é a cidade 5. Ela dista 7 unidades de um extremo da rota (cidade 2) e 2 unidades do outro extremo (cidade 4). Logo, a cidade 5 deve ser conectada à cidade 4, isto é, a rota corrente é: $s = (2 \to 6 \to 1 \to 3 \to 4 \to 5)$.

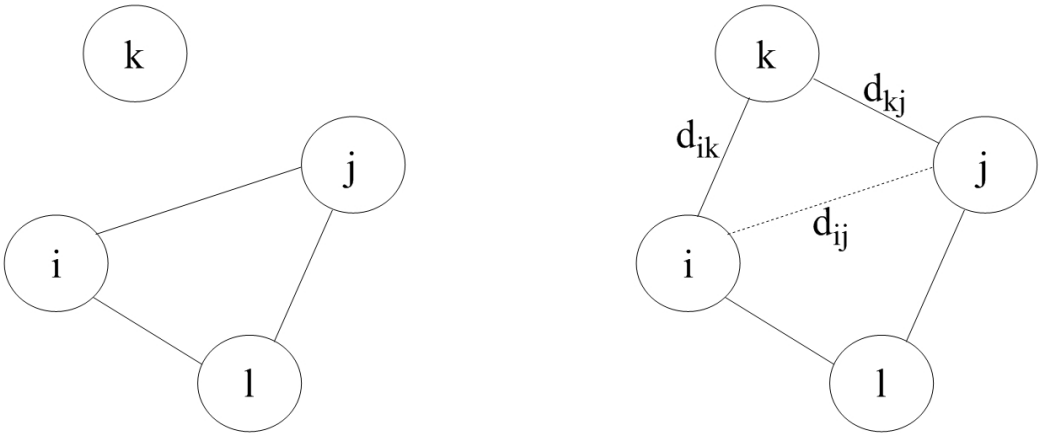
vi) Passo 6: Como todas as cidades já foram visitadas, resta agora somente conectar as duas extremidades (cidades 5 e 2) para formar um ciclo hamiltoniano.

Ao final desses 6 passos, teremos produzido a solução $s = (2\ \ 6\ \ 1\ \ 3\ \ 4\ \ 5)$. Para esta solução, a distância total percorrida é $dist = d_{26} + d_{61} + d_{13} + d_{34} + d_{45} + d_{52} = 2+1+1+3+2+7 = 16$.

(c) **Heurística da Inserção Mais Barata**:
Nesta heurística, parte-se de uma subrota inicial envolvendo três cidades e, a cada passo, adiciona-se uma cidade k ainda não visitada entre as cidades i e j da subrota cujo custo de inserção s_{ij}^k dado pela fórmula abaixo seja a menor possível.

$$s_{ij}^k = d_{ik} + d_{kj} - d_{ij}$$

As figuras a seguir ilustram a inserção da cidade k entre as cidades i e j.



(a) Antes da inserção (b) Depois da inserção

Observa-se que a subrota inicial pode ser formada por um procedimento construtivo qualquer. Por exemplo, parta da cidade origem e adicione à subrota a cidade mais próxima. A seguir, considerando as duas extremidades (cidade origem e última cidade inserida), adicione a cidade ainda não visitada cuja soma das distâncias às duas extremidades seja a menor.

No exemplo considerado, considerando-se a cidade 1 como a cidade origem, constrói-se uma solução com os seguintes passos:

i) Passo 1: Adicione a cidade 3 à rota, já que sua distância à cidade 1 é a menor (A cidade 6 também tem mesma distância, e também poderia ser escolhida).

ii) Passo 2: Das cidades ainda não visitadas (2, 4, 5 e 6), a cidade 2 é a aquela cuja distância às cidades extremas 1 e 3 é a menor, no caso, $d_{21} + d_{32} = 2 + 5 = 7$. Então, a cidade 2 é a escolhida e deve ser conectada às cidades 3 e 2, isto é, a subrota corrente é: $s = (1 \rightarrow 3 \rightarrow 2)$, com a cidade 2 ligada à cidade 1. Com os passos 2 e 3 encerra-se a construção de uma subrota inicial envolvendo três cidades. A distância total percorrida é: $d(s) = d_{13} + d_{32} + d_{21} = 1 + 5 + 2 = 8$.

iii) Passo 3: Das cidades ainda não visitadas (4, 5 e 6), calculemos o custo de inserção entre todas as cidades i e j da subrota. A tabela a seguir mostra os custos de inserção.

i	k	j	$s_{ij}^k = d_{ik} + d_{kj} - d_{ij}$
1	4	3	$s_{13}^4 = 4 + 3 - 1 = 6$
1	5	3	$s_{13}^5 = 9 + 8 - 1 = 16$
1	6	3	$s_{13}^6 = 1 + 6 - 1 = 6$
3	4	2	$s_{32}^4 = 3 + 9 - 5 = 7$
3	5	2	$s_{32}^5 = 8 + 7 - 5 = 10$
3	6	2	$s_{32}^6 = 6 + 2 - 5 = 3$
2	4	1	$s_{21}^4 = 9 + 4 - 2 = 11$
2	5	1	$s_{21}^5 = 7 + 9 - 2 = 14$
2	6	1	$s_{21}^6 = 2 + 1 - 2 = 1^*$

Como o menor custo de inserção é s_{21}^6, então a cidade 6 deve ser inserida entre as cidades 2 e 1. Logo, a subrota corrente passa a ser: $s = (1 \rightarrow 3 \rightarrow 2 \rightarrow 6)$. A distância associada a essa subrota é: $d(s) = d(s)_{\text{anterior}} + s_{21}^6 = 8 + 1 = 9$.

iv) Passo 4: Das cidades ainda não visitadas (4 e 5), calculemos o custo de inserção entre todas as cidades i e j da subrota corrente. A tabela a seguir mostra os custos de inserção.

i	k	j	$s_{ij}^k = d_{ik} + d_{kj} - d_{ij}$
1	4	3	$s_{13}^4 = 4 + 3 - 1 = 6^*$
1	5	3	$s_{13}^5 = 9 + 8 - 1 = 16$
3	4	2	$s_{32}^4 = 3 + 9 - 5 = 7$
3	5	2	$s_{32}^5 = 8 + 7 - 5 = 7$
2	4	6	$s_{26}^4 = 9 + 5 - 2 = 12$
2	5	6	$s_{26}^5 = 7 + 2 - 2 = 7$
6	4	1	$s_{61}^4 = 5 + 4 - 1 = 8$
6	5	1	$s_{61}^5 = 2 + 9 - 1 = 10$

Como o menor custo de inserção é s_{13}^4, então a cidade 4 deve ser inserida entre as cidades 1 e 3. Logo, a subrota corrente passa a ser: $s = (1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 6)$. A distância associada a essa subrota é: $d(s) = d(s)_{\text{anterior}} + s_{13}^4 = 9 + 6 = 15$.

v) Passo 5: A única cidade ainda não visitada é a cidade 5. A tabela a seguir mostra os custos de inserção dessa cidade entre todas as arestas da subrota corrente.

i	k	j	$s_{ij}^k = d_{ik} + d_{kj} - d_{ij}$
1	5	4	$s_{14}^5 = 9 + 2 - 4 = 7^*$
4	5	3	$s_{43}^5 = 2 + 8 - 3 = 7$
3	5	2	$s_{32}^5 = 8 + 7 - 5 = 10$
2	5	6	$s_{26}^5 = 7 + 2 - 2 = 7$
6	5	1	$s_{61}^5 = 2 + 9 - 1 = 10$

Como o menor custo de inserção é s_{14}^5, então a cidade 5 deve ser inserida entre as cidades 1 e 4. Logo, a rota resultante é: $s = (1 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 6)$. A distância associada a essa rota é: $d(s) = d(s)_{\text{anterior}} + s_{14}^5 = 15 + 7 = 22$.

5.3.2 Heurísticas de Refinamento

Uma heurística de refinamento consiste em promover modificações em uma solução de forma a tentar melhorá-la. Elas são baseadas na noção de vizinhança. Mais especificamente, seja S o espaço de busca de um problema de otimização e f a função objetivo a otimizar (minimizar ou maximizar). A função N, a qual depende do problema considerado, associa a cada solução $s \in S$ sua vizinhança $N(S) \subseteq S$. Cada solução $s' \in N(s)$ é chamada de vizinho de s. Denomina-se *movimento* a modificação m que transforma uma solução s em outra, s', que esteja em sua vizinhança. Representa-se essa operação por $s' \leftarrow s \oplus m$.

Em linhas gerais, nesta classe de heurísticas parte-se de uma solução inicial qualquer (a qual pode ser obtida por uma heurística construtiva ou então gerada aleatoriamente) e caminha-se, a cada iteração, de vizinho para vizinho de acordo com a definição de vizinhança adotada.

No caso do PCV, um movimento m pode ser a troca entre duas posições no vetor s. Assim, se $s = (4\ 3\ 1\ 2)^t$, então, com este tipo de movimento, são seus vizinhos as seguintes soluções: $s_1^{'} = (3\ 4\ 1\ 2)^t$, $s_2^{'} = (1\ 3\ 4\ 2)^t$, $s_3^{'} = (2\ 3\ 1\ 4)^t$, $s_4^{'} = (4\ 1\ 3\ 2)^t$, $s_5^{'} = (4\ 2\ 1\ 3)^t$

e $s_6^{'} = (4\ 3\ 2\ 1)^t$. No caso de o movimento m ser a realocação de uma cidade em outra posição na sequência, os vizinhos de s serão: $s_1^{'} = (3\ 4\ 1\ 2)^t$, $s_2^{'} = (3\ 1\ 4\ 2)^t$, $s_3^{'} = (4\ 1\ 3\ 2)^t$, $s_4^{'} = (4\ 1\ 2\ 3)^t$, $s_5^{'} = (4\ 3\ 2\ 1)^t$. Há outros movimentos mais elaborados, tal como o movimento Or, que consiste em realocar um bloco contíguo de cidades em outra posição da sequência. No exemplo considerado, considerando blocos de tamanho 2, teríamos os seguintes vizinhos: $s_1^{'} = (1\ 4\ 3\ 2)^t$, $s_2^{'} = (4\ 2\ 3\ 1)^t$, $s_3^{'} = (4\ 1\ 2\ 3)^t$. Neste exemplo, o primeiro vizinho é gerado pela inserção do bloco (4 3) entre as cidades 1 e 2; o segundo vizinho, pela inserção do bloco (3 1) entre as cidades 2 e 4 e, finalmente, o terceiro vizinho, pela inserção do bloco (1 2) entre as cidades 4 e 3.

Uma heurística clássica de refinamento é o Método da Descida (*Descent Method*) - no caso de o problema ser de minimização - ou Método da Subida (*Hill climbing Method*)- no caso de o problema ser de maximização. Nesse método parte-se de uma solução inicial qualquer e a cada passo são analisados todos os seus possíveis vizinhos, movendo-se somente para aquele que representar uma melhora no valor atual da função de avaliação. Desta forma, o método pára quando um ótimo local, com relação à vizinhança utilizada, é encontrado.

Uma variante do método da Descida/Subida é o Método de Primeira Melhora (*First Improvement Method*). Nesse método, interrompe-se a exploração da vizinhança quando um vizinho melhor é encontrado. Desta forma, apenas no pior caso, toda a vizinhança é explorada. Entretanto, tal como no método da descida/subida, este método também fica preso no primeiro ótimo local encontrado.

Outro método alternativo, que evita essa pesquisa exaustiva é o Método de Descida/Subida Randômica (*Random Descent/Uphill Method*). Ele consiste em analisar um vizinho qualquer e o aceitar somente se ele for estritamente melhor que a solução corrente; não o sendo, a solução corrente permanece inalterada e outro vizinho é gerado. O procedimento é interrompido após um número fixo de iterações sem melhora no valor da melhor solução obtida até então. Como nesse método não é feita a exploração de toda a vizinhança da solução corrente, não há garantia de que a solução final seja um ótimo local com relação à vizinhança adotada.

Na Figura 6 mostra-se o pseudocódigo do Método Randômico de Descida aplicado ao refinamento de uma solução s em um problema de minimização de uma função $f(.)$, utilizando uma estrutura de vizinhança $N(.)$. Nesta figura, *IterMax* representa o número máximo de iterações sem melhora no valor da função de avaliação.

```
procedimento DescidaRandomica(f(.), N(.), IterMax, s);
1  Iter ← 0;      {Contador de iterações sem melhora }
2  enquanto (Iter < IterMax) faça
3      Iter ← Iter + 1;
4      Selecione aleatoriamente s' ∈ N(s);
5      se (f(s') < f(s)) então
6          Iter ← 0;
7          s ← s' ;
8      fim-se;
9  fim-enquanto;
10 Retorne s;
fim DescidaRandomica;
```

Figura 6: Método de Descida Randômica

5.4 Variantes do PCV

5.4.1 Problema dos m-Caixeiros Viajantes

Nesta variante do PCV há m Caixeiros e se deseja minimizar a distância total percorrida por todos eles. Considere os seguintes parâmetros de entrada:

$Cidades$	:	Conjunto das cidades
d_{ij}	:	Distância da cidade i à cidade j
n	:	Cardinalidade do conjunto de cidades, isto é, $n = \|Cidades\|$
m	:	Número de Caixeiros Viajantes

(a) Variáveis de decisão:
x_{ij} variável binária que assume valor 1 se o arco (i, j) for utilizado e 0, caso contrário
f_{ij} quantidade de fluxo enviada da cidade i para a cidade j

(b) Função objetivo:
$\min \sum_{i \in Cidades} \sum_{j \in Cidades} d_{ij} x_{ij}$

(c) Restrições:

c.1) À cada cidade k, exceto a origem (cidade de índice 1), só chega um arco:
$\sum_{i \in Cidades} x_{ik} = 1 \;\; \forall k \in Cidades \,|\, k \neq 1$

c.2) De cada cidade k, exceto a origem (cidade de índice 1), só sai um arco:
$\sum_{j \in Cidades} x_{kj} = 1 \;\; \forall k \in Cidades \,|\, k \neq 1$

c.3) Da cidade origem saem m arcos:
$\sum_{j \in Cidades} x_{1j} = m$

c.4) À cidade origem chegam m arcos:
$\sum_{i \in Cidades} x_{i1} = m$

c.5) Exceto para a cidade origem (primeira cidade), o fluxo que chega a uma cidade k menos o que sai de k é igual a 1:
$\sum_{i \in Cidades} f_{ik} - \sum_{j \in Cidades} f_{kj} = 1 \;\; \forall k \in Cidades \mid k \neq 1$

c.6) O fluxo máximo que passa em um arco usado no percurso é inferior a $n - m$, onde n é o número de cidades e m é o número de caixeiros:
$f_{ij} \leq (n - m) x_{ij} \;\; \forall i \in Cidades, \;\; \forall j \in Cidades$

c.7) Integralidade e não-negatividade:
$x_{ij} \in \{0, 1\} \;\; \forall i \in Cidades, \;\; \forall j \in Cidades$
$f_{ij} \geq 0 \;\; \forall i \in Cidades, \;\; \forall j \in Cidades$

5.4.2 Problema do Caixeiro Viajante com Coleta Seletiva de Prêmios

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), referido na literatura inglesa como *Prize Collecting Traveling Salesman Problem* (PCTSP), é uma variante do Problema do

Caixeiro Viajante. O PCVCP pode ser associado a um caixeiro viajante que coleta um prêmio p_k, não negativo, em cada cidade k que ele visita e paga uma penalidade γ_k para cada cidade k que não visita, com um custo c_{ij} de deslocamento entre as cidades i e j. O problema encontra-se em minimizar o somatório dos custos da viagem e penalidades, enquanto inclui na sua rota um número suficiente de cidades que o permita coletar um prêmio mínimo, p_{min}, pré-estabelecido.

O PCVCP foi formulado inicialmente por Egon Balas [2] como um modelo para a programação da operação diária de uma fábrica que produzia lâminas de aço. Por razões que tinham a ver com o desgaste dos rolos e também por outros fatores, a sequência na ordem do processamento era essencial. A programação consistia na escolha de um número de lâminas associadas às suas ordens de execução, que satisfizessem o limite inferior do peso, e que ordenadas numa sequência apropriada, minimizasse a função de sequência. As tarefas de escolha das lâminas e das opções disponíveis para o seu sequenciamento necessitavam ser resolvidas em conjunto.

Para a modelagem de programação matemática do PCVCP, sejam os seguintes parâmetros de entrada:

$Cidades$	:	Conjunto das cidades
c_{ij}	:	Custo de deslocamento da cidade i à cidade j
n	:	Cardinalidade do conjunto de cidades, isto é, $n = \|Cidades\|$
p_k	:	Prêmio coletado por visitar a cidade k
γ_k	:	Penalidade paga por não visitar a cidade k

(a) Variáveis de decisão:
x_{ij} variável binária que assume valor 1 se o arco (i, j) for utilizado e 0, caso contrário

f_{ij} a quantidade de fluxo enviada da cidade i para a cidade j

z_k variável binária que assume valor 1 se a cidade k for visitada e 0, caso contrário

(b) Função objetivo:
$$\min \sum_{i \in Cidades} \sum_{j \in Cidades} c_{ij} x_{ij} + \sum_{k \in Cidades} \gamma_k (1 - z_k)$$

(c) Restrições:

c.1) À cada cidade k visitada só chega um arco:
$$\sum_{i \in Cidades} x_{ik} = z_k \ \ \forall k \in Cidades$$

c.2) De cada cidade k visitada só sai um arco:
$$\sum_{j \in Cidades} x_{kj} = z_k \ \ \forall k \in Cidades$$

c.3) O prêmio mínimo p_{min} deve ser coletado:
$$\sum_{k \in Cidades} p_k z_k \geq p_{min}$$

c.4) Exceto para a cidade origem (primeira cidade), o fluxo que chega a uma cidade k menos o que sai de k é igual a 1 se a cidade k for visitada e 0, caso contrário:
$$\sum_{i \in Cidades} f_{ik} - \sum_{j \in Cidades} f_{kj} = z_k \ \ \forall k \in Cidades \mid k \neq 1$$

c.5) O fluxo máximo que passa em um arco usado no percurso é inferior a $n - 1$, sendo n o número de cidades:
$$f_{ij} \leq (n-1) x_{ij} \ \ \forall i \in Cidades, \ \ \forall j \in Cidades$$

c.6) Integralidade e não-negatividade:
$x_{ij} \in \{0,1\} \ \ \forall i \in Cidades, \ \ \forall j \in Cidades$
$f_{ij} \geq 0 \ \ \forall i \in Cidades, \ \ \forall j \in Cidades$
$z_k \in \{0,1\} \ \ \forall k \in Cidades$

6 Problema de Roteamento de Veículos

6.1 Definição

Seja um conjunto de consumidores $\{1, 2, \cdots, n\}$ e uma frota ilimitada de veículos sediada em um único depósito 0. Para cada par (i, j) é dado o custo de ligação c_{ij}. No problema básico de roteamento de veículos, a frota é homogênea, isto é, os veículos têm a mesma capacidade (*cap*). O PRV consiste em encontrar as rotas de custo mínimo para os veículos satisfazendo as seguintes condições:

i) Toda rota começa e termina no depósito

ii) A demanda q_k de todos os consumidores deve ser atendida

iii) Em toda rota, a demanda q_k atendida não pode ultrapassar a capacidade *cap* do veículo

6.2 Modelagem de Programação Matemática

Considere os seguintes parâmetros de entrada:

V : Conjunto dos consumidores e o depósito, isto é, $V = \{0, 1, 2, \cdots, n\}$
c_{ij} : Custo de ligação entre os elementos i e j de V
q_k : Demanda do consumidor k. No caso do depósito, tem-se $q_0 = 0$
cap : Capacidade de cada veículo

(a) Variáveis de decisão:
x_{ij} variável binária que assume valor 1 se o arco (i, j) for utilizado e 0, caso contrário
f_{ij} a quantidade de fluxo enviada do nó i para o nó j

(b) Função objetivo:
$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$

(c) Restrições:

c.1) À cada nó k, exceto aquele referente ao depósito 0, só chega um arco:
$\sum_{i \in V} x_{ik} = 1 \ \ \forall k \in V, \ k \neq 0$

c.2) De cada nó k, exceto aquele referente ao depósito 0, só sai um arco:
$\sum_{j \in V} x_{kj} = 1 \ \ \forall k \in V, \ k \neq 0$

c.3) No depósito 0, o número de arcos que saem é igual ao número de arcos que chegam:
$\sum_{j \in V} x_{0j} = \sum_{i \in V} x_{i0}$

c.4) Exceto para o nó referente ao depósito, o fluxo que chega ao nó k menos o que sai de k é igual à demanda associada ao k-ésimo nó:
$\sum_{i \in V} f_{ik} - \sum_{j \in V} f_{kj} = q_k \ \ \forall k \in V \mid k \neq 0$

c.5) O fluxo máximo que passa em um arco usado no percurso é inferior a *cap*:
$f_{ij} \leq (cap) x_{ij} \ \ \forall i \in V, \ \ \forall j \in V$

c.6) Integralidade e não-negatividade:
$x_{ij} \in \{0, 1\} \ \ \forall i \in V, \ \ \forall j \in V$
$f_{ij} \geq 0 \ \ \forall i \in V, \ \ \forall j \in V$

Uma formulação de eliminação de subciclos que requer menor quantidade de variáveis é apresentada a seguir. Estas restrições substituem as de fluxo apresentadas anteriormente (restrições c.4 e c.5):

$u_k \geq u_i + q_k - cap + cap \times (x_{ki} + x_{ik}) - (q_k + q_i) \times x_{ki} \ \ \forall k \in V, k \neq 0, \forall i \in V, \ i \neq 0, \ i \neq k$
$q_k \leq u_k \leq cap \ \ \forall k \in V, k \neq 0$
$u_k \leq cap - (cap - q_k) \times x_{0k} \ \ \forall k \in V, k \neq 0$
$u_k \geq q_k + \sum_{i \in V \mid i \neq 0} (q_i \times x_{ik}) \ \ \forall k \in V, k \neq 0$

Observamos que 0, nessas expressões, representa o depósito. Assim, no LINGO, devemos aludí-lo a $@index(0)$ ou, simplesmente, ao cliente de índice 1 (considerando que o mesmo é o primeiro da lista).

A seguir, são apresentados dois modelos LINGO referentes ao PRV, em que V $= \{0, 1, 2, \cdots, n\}$. O primeiro interfaceia com uma planilha Excel, enquanto no segundo a leitura de dados é feita a partir de um arquivo txt. No primeiro modelo, escreve-se $@index(0)$ para se referenciar ao depósito (0) e são usadas variáveis de fluxo para eliminar subciclos. Já no segundo modelo, pressupõe-se que o depósito é a primeira cidade do conjunto V; assim, tem índice 1.

```
model:
sets:
  cidades / @ole('prv.xls','cidades')/: q;
  matriz(cidades, cidades): c,  ! Matriz de custos;
                            x,  ! x(i, j) = 1 se o arco (i,j) fizer parte da solução;
                            f;  ! Fluxo de i para j;
endsets

data:
  q, cap = @ole('prv.xls','demanda','capVeic');
enddata

[fo]  min = @sum(matriz(i,j): c(i,j)*x(i,j));

! De cada cidade k, exceto o depósito, só sai um único veículo;
@for(cidades(k) | k #NE# @index(0): @sum(cidades(j): x(k,j)) = 1);

! A cada cidade k, exceto o depósito, só chega um único veículo;
@for(cidades(k) | k #NE# @index(0): @sum(cidades(i): x(i,k)) = 1);

! O número de veículos que saem do depósito deve ser igual
  ao número de veículos que chegam ao depósito;
```

```
@sum(cidades(j): x(@index(0), j)) = @sum(cidades(i): x(i, @index(0)));

! Ao passar por uma cidade k, exceto o depósito (0), o veículo deve atender a
  demanda dessa cidade, i.é, deve deixar q(k) unidades de produto na cidade k;
@for(cidades(k) | k #ne# @index(0):
     @sum(cidades(i): f(i,k)) - @sum(cidades(j): f(k,j) ) = q(k) );

! O fluxo máximo em cada aresta não pode superar a capacidade do veículo;
@for(matriz(i,j): f(i,j) <= (cap)*x(i,j));

! As variáveis x são binárias;
@for(matriz(i,j): @bin(x(i,j)));

! Exporta a solução para o arquivo prv.xls;
data:
  @ole('prv.xls','x','fo') = x, fo;
enddata

end
```

O segundo modelo requer um menor número de variáveis e utiliza variáveis reais $u \geq 0$ para eliminar subciclos. Como dito anteriormente, considera-se que o depósito é o primeiro elemento do conjunto V. Além disso, as distâncias entre os elementos de V são calculadas a partir de suas coordenadas. Nesse modelo, a entrada de dados é via arquivo txt.

```
! Cidades;
Dep A B C D E F G H I J K ~

! Coordenadas x;
30 37 49 52 20 40 21 17 31 52 51 42  ~

! Coordenadas y;
40 52 49 64 26 30 47 63 62 33 21 41  ~

! Demanda dos clientes;
 0 14 26 18 26 32 17 27  8 16 15 28  ~

! Capacidade dos veículos;
50
```

```
model:
title: Problema de Roteamento de Veículos;
sets:
  V / @file('PRV.txt') /: u, q, coord_x, coord_y;
  Matriz(V, V): d, x;
endsets

! Leitura dos dados;
data:
  coord_x = @file('PRV.txt');
```

```
  coord_y = @file('PRV.txt');
  q = @file('PRV.txt');
  cap = @file('PRV.txt');
enddata

! Cálculo das distâncias entre os elementos de V;
@for(V(i):
   x(i,i) = 0;
   @for(V(j):
      d(i,j) = ( (coord_x(j) - coord_x(i))^2 +
                 (coord_y(j) - coord_y(i))^2   )^(0.5)));

[fo] min = @sum(V(i): @sum(V(j): d(i,j) * x(i,j)));

! A cada cidade k, exceto o depósito, só chega um arco;
@for(V(k) | k #ne# 1:
   @sum(V(i): x(i,k)) = 1);

! De cada cidade k, exceto o depósito, só sai um arco;
@for(V(k) | k #ne# 1:
   @sum(V(j): x(k,j)) = 1);

! O número de veículos que saem do depósito deve ser igual
  ao número de veículos que chegam ao depósito;
@sum(cidades(j): x(1, j)) = @sum(cidades(i): x(i, 1));

! Restrições de eliminação de subciclos;
@for(V(k):
   @for(V(i) | i #ne# k #and# i #ne# 1:
       u(k) >= u(i) + q(k) - cap + cap*(x(k,i) + x(i,k)) - (q(k) + q(i))*x(k,i)));

@for(V(k) | k #NE# 1:
   @bnd(q(k), u(k), cap));

@for(V(k) | k #NE# 1:
   u(k) <= cap - (cap - q(k))*x(1,k));

@for(V(k) | k #NE# 1:
   u(k) >= q(k) + @sum(V(i) | i #ne# 1: q(i)*x(i,k)));

! As variáveis de decisão x são binárias;
@for(V(i):
   @for(V(j): @bin(x(i,j))));

end
```

6.3 Geração de colunas para o PRV

Seja $Colunas = \{1, 2, \cdots, n\}$ um conjunto de colunas (rotas) para o PRV e $Linhas = \{1, 2, \cdots, m\}$ um conjunto de clientes a serem visitados (linhas). Seja c_j o custo da coluna j.

O PRV, nesse caso, consiste em escolher as colunas de tal forma que cada linha seja coberta por uma única coluna a custo mínimo.

Para essa modelagem de programação matemática, conhecida como modelo de particionamento de conjuntos, sejam os seguintes parâmetros de entrada:

$Colunas$	:	Conjunto das colunas (rotas)
$Linhas$	:	Conjunto das linhas (clientes a serem visitados)
c_j	:	Custo da coluna j
a_{ij}	:	Parâmetro que assume o valor 1 se o cliente i for atendido pela rota j e 0, c.c.

(a) Variáveis de decisão:
x_j variável binária que assume valor 1 se a coluna j for utilizada e 0, caso contrário

(b) Função objetivo:
$\min \sum_{j \in Colunas} c_j x_j$

(c) Restrições:

c.1) Cada linha i é coberta por uma única coluna j:
$\sum_{j \in Colunas} a_{ij} x_j = 1 \ \ \forall i \in Linhas$

c.2) Integralidade e não-negatividade:
$x_j \in \{0, 1\} \ \ \forall j \in Colunas$

Segue um pequeno exemplo com 8 clientes e 6 possíveis rotas.

	Rotas					
Clientes	1	2	3	4	5	6
1	1	0	0	0	0	0
2	1	0	0	0	0	0
3	1	0	0	0	0	0
4	0	1	0	0	1	0
5	0	1	0	0	0	1
6	0	0	1	0	1	0
7	0	0	1	0	0	1
8	0	0	0	1	1	0
Custo	100	50	70	60	80	40

Nesse exemplo, a rota 1 atende aos clientes 1, 2 e 3 ao custo de 100 unidades monetárias. Já a rota 2 atende aos clientes 4 e 5 ao custo de 50 u.m.

Observe que na solução de particionamento de conjuntos, cada linha deve ser coberta por uma única coluna. Assim, por exemplo, o cliente 4 não pode ser atendido conjuntamente pelas rotas 2 e 4.

Uma solução para esse exemplo é: $x^{(1)} = (1\ 1\ 1\ 1\ 0\ 0)^t$, cujo custo é $f(x^{(1)}) = 100 + 50 + 70 + 60 = 280$. Nesse vetor $x^{(1)}$ fazem parte da solução as colunas 1, 2, 3 e 4. Outra possível solução é: $x^{(2)} = (1\ 0\ 0\ 0\ 1\ 1)^t$, cujo custo é $f(x^{(2)}) = 100 + 80 + 40 = 220$. Essa segunda solução é composta pelas colunas 1, 5 e 6. O que se deseja é determinar quais colunas fazem parte da solução ótima.

O PRV também pode ser modelado como um problema de recobrimento de conjuntos, conhecido na literatura inglesa como *Set Covering Problem*. Neste caso, cada cliente deve ser coberto por pelo menos uma rota (coluna). O modelo de programação matemática correspondente é:

(a) Variáveis de decisão:
x_j variável binária que assume valor 1 se a coluna j for utilizada e 0, caso contrário

(b) Função objetivo:
$\min \sum_{j \in Colunas} c_j x_j$

(c) Restrições:

c.1) Cada linha i é coberta por pelo menos uma coluna j:
$\sum_{j \in Colunas} a_{ij} x_j \geq 1 \ \ \forall i \in Linhas$

c.2) Integralidade e não-negatividade:
$x_j \in \{0, 1\} \ \ \forall j \in Colunas$

Nesta última modelagem, se um cliente porventura for visitado (coberto) por mais de uma rota, então, na solução final, deve-se escolher sua ocorrência em apenas uma das rotas e eliminá-lo das demais.

6.4 Modelos Heurísticos para o PRV

6.4.1 Heurísticas Construtivas

(a) **Adaptação da Heurística do Vizinho Mais Próximo ao PRV**

Nesta heurística, a ideia é começar com um veículo no depósito e ir para o cliente mais próximo que ainda possa ser visitado sem desrespeitar as restrições do problema. Caso o veículo não possa atender mais clientes, deve-se retornar ao depósito e recomeçar o procedimento com outro veículo. O procedimento pára quando todos os clientes forem atendidos.

Exemplo: Considere a matriz de custos a seguir, onde o depósito é referenciado pelo número 0, e as demandas de cada uma das 5 cidades. Sabendo que os veículos têm 20 unidades de capacidade, determine as rotas de custo mínimo para os veículos.

Cliente	0	1	2	3	4	5	Demanda
0	0	6	7	8	9	10	0
1	6	0	3	2	1	4	5
2	7	3	0	5	3	4	9
3	8	2	5	0	8	1	6
4	9	1	3	8	0	5	4
5	10	4	4	1	5	0	7

Na sequência de passos a seguir, mostra-se como construir uma solução para o PRV pela Heurística do Vizinho Mais Próximo partindo-se do depósito.

i) Passo 0: Sai-se do depósito com o veículo 1.

ii) Passo 1: Adicione a cidade 1 à rota do primeiro veículo, já que sua distância ao depósito é a menor ($d_{10} = 6 < d_{i0} \ \ \forall i \in V$) e a demanda acumulada (5 unidades) é menor que a capacidade do veículo (20 unidades).

iii) Passo 2: Adicione a cidade 4 à rota do primeiro veículo, já que sua distância à última cidade visitada (cidade 1) é a menor dentre as cidades ainda não visitadas (no caso, as cidades 2, 3, 4 e 5), e a demanda acumulada (9 unidades) é menor que a capacidade do veículo (20 unidades).

iv) Passo 3: Adicione a cidade 2 à rota do primeiro veículo, já que sua distância à última cidade visitada (cidade 4) é a menor dentre todas as cidades ainda não visitadas (no caso, as cidades 2, 3 e 5) e a demanda acumulada (18 unidades) é menor que a capacidade do veículo (20 unidades).

v) Passo 4: A cidade mais próxima à cidade 2, dentre as ainda não visitadas (cidades 3 e 5), é a cidade 5. No entanto, a demanda dessa cidade (7 unidades) não pode ser atendida pelo veículo 1, pois seria ultrapassada a capacidade do veículo 1. Passa-se então para a segunda cidade mais próxima, no caso, a cidade 3. Também nesse caso sua demanda (6 unidades) não pode ser atendida pelo veículo 1. Como as cidades não visitadas não podem ser atendidas pelo veículo 1, retorna-se ao depósito, fechando-se a rota do primeiro veículo.

vi) Passo 5: Sai-se do depósito com o veículo 2.

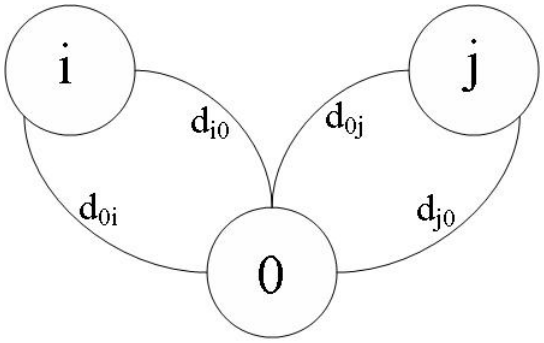
vii) Passo 6: Adicione a cidade 3 à rota do segundo veículo, já que sua distância ao depósito é a menor dentre as duas cidades ainda não visitadas (cidades 3 e 5), pois $d_{03} = 8 < d_{05} = 10$) e a demanda acumulada (6 unidades) pode ser atendida.

viii) Passo 7: A única cidade ainda não visitada é a cidade 5. Sua demanda, de 7 unidades, pode ser atendida pelo veículo corrente, pois a demanda acumulada passa a ser de 13 unidades, que é menor que a capacidade do veículo 2 (de 20 unidades).

ix) Passo 8: Não há cidades não atendidas. Logo, deve-se retornar ao depósito com o segundo veículo.

Ao final desses 8 passos, teremos produzido as seguintes rotas:
Rota 1: $s^{(1)} = (0 \quad 1 \quad 4 \quad 2)$. Nesta rota, a distância percorrida pelo primeiro veículo é $dist(s^{(1)}) = d_{01} + d_{14} + d_{42} + d_{20} = 6 + 1 + 3 + 7 = 17$. A carga útil do veículo 1 é de 18 unidades, ou 90% de sua capacidade.
Rota 2: $s^{(2)} = (0 \ \ 3 \ \ 5)$. Nesta rota, a distância percorrida pelo segundo veículo é $dist(s^{(2)}) = d_{03} + d_{35} + d_{50} = 8 + 1 + 10 = 19$. A carga útil do veículo 2 é de 13 unidades, ou 65% de sua capacidade.
O número de veículos utilizados é, portanto, 2 e a distância total percorrida por eles é $dist = dist(s^{(1)}) + dist(s^{(2)}) = 17 + 19 = 36$ unidades de distância.

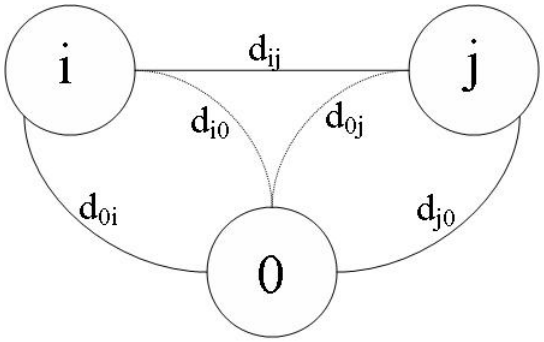
(b) **Heurística de Clarke e Wright**
Este método começa com um veículo atendendo um cliente e retornando ao depósito. A figura a seguir ilustra essa situação, onde se mostram duas rotas, uma atendendo a cidade i e a outra à cidade j.



A seguir, são feitas todas as possíveis combinações entre duas rotas de modo que um veículo possa ser eliminado e a distância de viagem, reduzida. Isto é, deve ser calculada a economia s_{ij} entre todos os pares (i, j) de cidades onde i é uma cidade da extremidade de uma rota e j uma extremidade de uma outra rota, conforme equação a seguir.

$$s_{ij} = d_{i0} + d_{0j} - d_{ij}$$

A figura seguinte ilustra a junção das duas rotas, uma envolvendo a cidade i e a outra, a cidade j.



É importante observar que as combinações de rotas são feitas apenas entre as cidades das extremidades das rotas. Além disso, só podem ser combinadas rotas que atendam às restrições de capacidade dos veículos envolvidos (e outras restrições porventura existentes, como por exemplo, janelas de tempo).

Calculadas todas as possíveis combinações (tarefa que é executada uma única vez), é realizada aquela combinação que produz a maior economia possível satisfazendo, naturalmente, as restrições estabelecidas.

Exemplo: Resolva o exemplo anterior pela Heurística de Clarke e Wright.

1^a Iteração:
Inicialmente, alocamos um veículo para atender cada um dos clientes. A distância total percorrida é:
$dist = d_{01}+d_{10}+d_{02}+d_{20}+d_{03}+d_{30}+d_{04}+d_{40}+d_{05}+d_{50} = 6+6+7+7+8+8+9+9+10+10 = 80.$
Número de veículos = 5.

i	j	d_{i0}	d_{0j}	d_{ij}	$s_{ij} = d_{i0} + d_{0j} - d_{ij}$	Demanda acumulada
1	2	6	7	3	$s_{12} = 6 + 7 - 3 = 10$	14
1	3	6	8	2	$s_{13} = 6 + 8 - 2 = 12$	11
1	4	6	9	1	$s_{14} = 6 + 9 - 1 = 14$	9
1	5	6	10	4	$s_{15} = 6 + 10 - 4 = 12$	12
2	3	7	8	5	$s_{23} = 7 + 8 - 5 = 10$	15
2	4	7	9	3	$s_{24} = 7 + 9 - 3 = 13$	13
2	5	7	10	4	$s_{25} = 7 + 10 - 4 = 13$	16
3	4	8	9	8	$s_{34} = 8 + 9 - 8 = 9$	10
3	5	8	10	1	$s_{35} = 8 + 10 - 1 = 17$ *	13
4	5	9	10	5	$s_{45} = 9 + 10 - 5 = 14$	11

* Maior economia

Como $s_{35} = \max\{s_{ij}\}\ \forall(i,j)$ e a soma das demandas das rotas envolvendo os clientes 3 e 5 não supera a capacidade de um veículo, devemos combinar essas duas rotas. A distância total percorrida é:
$dist = dist - s_{35} = 80 - 17 = 63$.
Número de veículos: 4

2ª Iteração:
Atualizemos o quadro anterior apenas com relação às demandas acumuladas, já que as economias serão as mesmas.

i	j	d_{i0}	d_{0j}	d_{ij}	$s_{ij} = d_{i0} + d_{0j} - d_{ij}$	Demanda acumulada
1	2	6	7	3	$s_{12} = 6 + 7 - 3 = 10$	14
1	3	6	8	2	$s_{13} = 6 + 8 - 2 = 12$	18
1	4	6	9	1	$s_{14} = 6 + 9 - 1 = 14$*	9
1	5	6	10	4	$s_{15} = 6 + 10 - 4 = 12$	18
2	3	7	8	5	$s_{23} = 7 + 8 - 5 = 10$	22
2	4	7	9	3	$s_{24} = 7 + 9 - 3 = 13$	13
2	5	7	10	4	$s_{25} = 7 + 10 - 4 = 13$	22
3	4	8	9	8	$s_{34} = 8 + 9 - 8 = 9$	17
4	5	9	10	5	$s_{45} = 9 + 10 - 5 = 14$	17

* Maior economia

Como $s_{14} = \max\{s_{ij}\}\ \forall(i,j)$ e a soma das demandas das rotas envolvendo os clientes 1 e 4 não supera a capacidade de um veículo (que é de 20 unidades), devemos combinar as rotas envolvendo os clientes 1 e 4. A distância total percorrida é:
$dist = dist - s_{14} = 63 - 14 = 49$.
Número de veículos: 3

3ª Iteração:
Igualmente, atualizemos o quadro anterior apenas com relação às demandas acumuladas, já que as economias serão as mesmas.

i	j	d_{i0}	d_{0j}	d_{ij}	$s_{ij} = d_{i0} + d_{0j} - d_{ij}$	Demanda acumulada
1	2	6	7	3	$s_{12} = 6 + 7 - 3 = 10$	18
1	3	6	8	2	$s_{13} = 6 + 8 - 2 = 12$	22
1	5	6	10	4	$s_{15} = 6 + 10 - 4 = 12$	22
2	3	7	8	5	$s_{23} = 7 + 8 - 5 = 10$	22
2	4	7	9	3	$s_{24} = 7 + 9 - 3 = 13$*	18
2	5	7	10	4	$s_{25} = 7 + 10 - 4 = 13$	22
3	4	8	9	8	$s_{34} = 8 + 9 - 8 = 9$	22
4	5	9	10	5	$s_{45} = 9 + 10 - 5 = 14$	22

* Maior economia

Como $s_{24} = \max\{s_{ij}\}\ \forall(i,j)$ e a soma das demandas das rotas envolvendo os clientes 2 e 4 não supera a capacidade de um veículo (que é de 20 unidades), devemos combinar as rotas envolvendo os clientes 2 e 4. A distância total percorrida é:
$dist = dist - s_{24} = 49 - 13 = 36$.
Número de veículos: 2

4^a Iteração:
Atualizemos o quadro anterior com relação às demandas acumuladas, eliminando as combinações já efetuadas, bem como aquelas que não têm as cidades i e j nas extremidades das rotas (no caso, as combinações envolvendo as cidades 1 e 2, 2 e 4, 3 e 4, e 4 e 5).

i	j	d_{i0}	d_{0j}	d_{ij}	$s_{ij} = d_{i0} + d_{0j} - d_{ij}$	Demanda acumulada
1	3	6	8	2	s_{13} = 6 + 8 - 2 = 12	31
1	5	6	10	4	s_{15} = 6 + 10 - 4 = 12	31
2	3	7	8	5	s_{23} = 7 + 8 - 5 = 10	31
2	5	7	10	4	s_{25} = 7 + 10 - 4 = 13	31

Como todas as combinações de duas rotas resulta em uma rota inviável, pois a demanda acumulada supera a capacidade de um veículo, o método pára e retorna como solução final as rotas:
Rota 1: $0 \to 1 \to 4 \to 2 \to 0$
Rota 2: $0 \to 3 \to 5 \to 0$
Distância total = 36
Número de veículos utilizados: 2

Tal como anteriormente, podem ser calculadas as taxas de utilização de cada veículo, as quais, no caso, são as mesmas da solução gerada pela Heurística do Vizinho Mais Próximo.

6.4.2 Heurísticas de refinamento

As heurísticas de refinamento clássicas do PRV são baseadas em movimentos envolvendo troca (*exchange* ou *swap*) e realocação (*insertion* ou *shift*) de clientes em uma mesma rota ou em rotas distintas.

Comentemos o funcionamento dessas heurísticas. Dada uma solução inicial s, obtida por uma heurística construtiva, são analisados todos os vizinhos possíveis usando-se o movimento de realocação (ou troca), inicialmente com realocações (trocas) intra-rotas e depois com realocações (trocas) inter-rotas. Se o melhor vizinho s' for melhor que a solução corrente s então s' passa a ser a nova solução corrente, isto é, $s \leftarrow s'$ e o procedimento continua a partir de s ; caso contrário, o procedimento retorna s como solução ótima local.

7 Enumeração Implícita em Programação Inteira 0-1

[Baseado no livro [5], dos professores Nelson Maculan Filho e Márcia Costa Fampa da UFRJ]

Seja:

$$\text{(P):}\quad \min\ z = \sum_{j=1}^{n} c_j x_j \qquad (1)$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad \forall i = 1, 2, \cdots, m \qquad (2)$$

$$x_j \in \{0,1\} \qquad \forall j = 1, 2, \cdots, n \qquad (3)$$

onde $c_j, a_{ij}, b_i \in \mathbb{R}$

Sem perda de generalidade, podemos sempre supor $c_j \geq 0 \ \forall j = 1, 2, \cdots, n$, pois no caso de existir k tal que $c_k < 0$, basta criar uma variável $y_k \in \{0, 1\}$ tal que $x_k = 1 - y_k$. Com essa mudança de variável, a nova função objetivo passa a ser $\sum_{\substack{j=1 \\ j \neq k}}^{n} c_j x_j - c_k x_k + c_k$. Assim, teremos que minimizar $z - c_k = \sum_{\substack{j=1 \\ j \neq k}}^{n} c_j x_j - c_k y_k$. Logo, a seguinte hipótese é assumida ao longo desta seção:

Hipótese: $c_j \geq 0 \ \ \forall j = 1, 2, \cdots, n$

Definimos também $c = (c_1 \ c_2 \ \cdots \ c_n)$, $b^t = (b_1 \ b_2 \ \cdots \ b_m)$, $x^t = (x_1 \ x_2 \ \cdots \ x_n)$ e $A = (a_{ij})_{m \times n}$ uma matriz com m linhas e n colunas.

O problema (P) poderá ser escrito também da seguinte maneira:

$$\begin{array}{llrlll} \text{(P):} & \min & \text{z} & = & cx & (4) \\ & & & & Ax \leq b & (5) \\ & & & & x \in \{0,1\}^n & (6) \end{array}$$

Uma solução de (3) ou (6) será representada por $x^p = (x_1^p \ x_2^p \ \cdots \ x_n^p)^t$, ou também pelo conjunto $J_p = \{j \mid x_j = 1\}$.

Exemplo: Para $x^7 = (0\ 1\ 1\ 1\ 0\ 0\ 1\ 1)^t$ tem-se $J_7 = \{2, 3, 4, 7, 8\}$.

Uma solução x^q é dita descendente de x^p se $J_p \subset J_q$. Exemplo: a solução $x^q = (1\ 1\ 1\ 0\ 1\ 1)^t$ é descendente da solução $x^p = (0\ 0\ 1\ 0\ 1\ 1)^t$.

Em alguns problemas, certas variáveis devem ser fixadas a priori para que possamos satisfazer as desigualdades de (2). Exemplos:

$3x_1 + 7x_2 + 2x_3 + x_4 \leq 5$ implica que $x_2 = 0$.

$2x_1 + 8x_2 + x_3 + x_4 \geq 9$ implica que $x_2 = 1$.

No caso em que estas duas restrições façam parte de (2), podemos assegurar que o conjunto de soluções viáveis é vazio.

Propriedade 1 Se x^p é uma solução de (3) então $\sum_{j \in J_p} c_j x_j \leq \sum_{j \in J_q} c_j x_j$ para todas as soluções x^q descendentes de x^p.

Demonstração:
Basta lembrar que $c_j \geq 0 \ \ \forall j = 1, 2, \cdots, n$ e que $J_p \subset J_q$.

Propriedade 2 Se $x^0 = (0\ 0\ \cdots\ 0)^t$ satisfaz (2), então x^0 é uma solução ótima de (P).

Demonstração:

Como, por hipótese, $c_j \geq 0$ e $x_j \in \{0, 1\}$ então $z = \sum_{j=1}^{n} c_j x_j \geq 0$. Logo, o valor mínimo de $z \geq 0$ é $z = 0$, ou seja, o ponto ótimo (ponto de mínimo) ocorre para $x = 0$.

Será apresentado, a seguir, um esquema de enumeração que supõe que a solução inicial x^0 é tal que $J_0 = \{\}$, isto é, $x^0 = (0\ 0\ \cdots\ 0)^t$ não satisfaz o conjunto de restrições (2). Se esse conjunto de restrições fosse satisfeito, então x^0 seria ótimo do problema (P) pela Propriedade 2.

Esquema de Enumeração:

Suponhamos que estejamos na solução x^p de (3) e que $\bar{z}$ seja a melhor solução viável de (P) encontrada até o momento, isto é, existe x^q tal que $\bar{z} = \sum_{j \in J_q} c_j x_j$ e que x^q seja viável de (P). Caso não tenhamos ainda encontrado uma solução viável, colocamos $\bar{z} = +\infty$.

A partir da solução x^p desejamos obter x^q descendente de x^p tal que $|J_q| = |J_p| + 1$ ou, equivalentemente, $J_q = J_p \cup \{l\}$, onde l é o índice da variável que assumirá valor unitário. Consideremos as seguintes hipóteses:

(i) Se x^p é viável de (P) então não nos interessa buscar um descendente x^q de x^p, pois $\sum_{j \in J_p} c_j x_j \leq \sum_{j \in J_q} c_j x_j$ (vide Propriedade 1).

(ii) Se $\sum_{j \in J_p} c_j x_j + c_l \geq \bar{z} \ \forall l \notin J_p$, isto quer dizer que todas as soluções descendentes de x^p fornecerão valores à função objetivo (1) sempre superiores ou iguais a $\bar{z}$. Nesse caso, também não nos interessa enumerar os descendentes de x^p.

(iii) Se existir índice i tal que:
$b_i - \sum_{j \in J_p} a_{ij} x_j - \sum_{j \notin J_p} \text{mínimo}\{0, a_{ij}\} < 0$
então nunca haverá um descendente de x^p viável. Assim, também não haverá interesse em enumerar os descendentes de x^p.

Definimos (i), (ii) e (iii) como **condições de parada** na solução x^p.

Quando tivermos parado em x^p por uma das três condições, teremos **enumerado implicitamente** todas as soluções descendentes de x^p.

Caso em x^p não tenhamos nenhuma condição de parada satisfeita, teremos que procurar uma solução descendente de x^p, por exemplo, x^q, tal que $J_q = J_p \cup \{l\}$, onde evidentemente $l \notin J_p$.

Consideremos $s_i = b_i - \sum_{j=1}^{n} a_{ij} x_j$ e $s_i \geq 0 \ \ \forall i = 1, 2, \cdots, m$, isto é, a variável s_i será de folga da restrição i.

Seja $s_i^p = b_i - \sum_{j \in J_p} a_{ij} x_j$, isto é, s_i^p representa o valor de s_i quando $x = x^p$.

Em x^p podemos definir os seguintes conjuntos:

1. $A_p = \{k \mid \sum_{j \in J_p} c_j x_j + c_k \geq \bar{z}, \ \ \forall k \notin J_p\}$

2. $D_p = \{k \mid \forall i \text{ com } s_i^p < 0, \text{ tem-se } a_{ik} \geq 0, \ \forall k \notin J_p\}$

3. $C_p = \{1, 2, \cdots, n\} - (J_p \cup A_p \cup D_p)$

Dito de outra forma, o conjunto D_p pode ser determinado verificando-se quais são as colunas k, dentre aquelas não pertencentes à solução parcial J_p, para as quais têm-se coeficientes a_{ik} maiores ou iguais a zero em toda a coluna. Para formar o conjunto A_p é necessário que <u>todas</u> as colunas k não pertencentes à solução J_p resultem em piora (ou manutenção) do valor da melhor solução encontrada até o momento, no caso, $\bar{z}$.

O conjunto C_p fornecerá os índices das variáveis candidatas a tomarem valor igual a 1, isto é, os índices $l \in C_p$ para formar uma solução x^q descendente de x^p na forma $J_q = J_p \cup \{l\}$. Observe que o conjunto A_p fornece os índices das variáveis que pioram (ou mantém) o valor da solução corrente caso entrem na solução. Já D_p reúne os índices das variáveis cuja entrada produzirá descendentes inviáveis. Assim, tanto A_p quanto D_p reunem índices para os quais não vale a pena incorporá-los à solução.

Seja $d_j^p = \sum_{i=1}^{m} \min\{0, s_i^p - a_{ij} x_j\}, \ j \in C_p$ e $d_l^p = \max_{j \in C_p}\{d_j^p\}$, isto é, d_l^p é a menor soma das inviabilidades. Em outras palavras, cada parcela $\min\{0, s_i^p - a_{ij} x_j\}$ representa o nível de inviabilidade existente na restrição i devido a inserção da variável x_j (com valor igual a 1) na solução

corrente, uma vez que somente os valores negativos de $s_i^p - a_{ij}x_j$ interessam. O somatório das inviabilidades de todas as restrições devido a inserção da variável que está na coluna j mede o nível de inviabilidade devido a esta variável. Quando se utiliza o máximo dos somatórios, o que se deseja é saber qual a coluna que consegue reduzir a inviabilidade ao menor nível.

Caso $d_l^p = 0$ então a solução descendente associada a $J_q = J_p \cup \{l\}$ será viável do problema (P).

No caso de haver mais de um índice para o qual $d_j^p = 0$, isto é,

$$L_p = \{j \in C_p \mid d_j^p = 0\}$$

então o índice l escolhido para a solução descendente será aquele associado a $c_l = \min_{j \in L_p}\{c_j\}$.

Obviamente, no caso em que nenhuma condição de parada seja verificada, tem-se $C_p \neq \{\}$.

Suponhamos, agora, que em x^q descendente direto de x^p, isto é, o último a ser desenvolvido a partir de x^p, o conjunto C_q seja vazio, ou ainda uma das três condições de parada seja satisfeita. Teremos, então, de x^q voltar a x^p e atualizar C_p de duas maneiras:

(i) $C_p = C_p - \{l\}$, onde l é tal que $J_q = J_p \cup \{l\}$.

(ii) A_p poderá ser modificado caso $\bar{z}$ também o seja, acarretando outra modificação em C_p.

O retorno de x^q a x^p é denominado *backtracking*. A enumeração pára completamente quando $C_0 = \{\}$. Deve ser observado que $\bar{z}$ é sempre atualizado ao se encontrar uma solução viável melhor que as anteriores. Caso o problema (P) seja vazio, $\bar{z} = \infty$ no final da enumeração.

Apresentaremos, a seguir, uma maneira de enumeração implícita finita, isto é, nunca enumeraremos explicitamente a mesma solução e, assim, a enumeração termina.

Usaremos uma estrutura de pilha proposta por Glover (1965) e Geoffrion (1967). Essa pilha representa o conjunto dos índices associados às variáveis fixadas.

Seja a pilha π, para a qual $p(j)$ será sua j-ésima componente tal que:

$$p(j) > 0 \text{ se } x_{p(j)} = 1$$
$$p(j) < 0 \text{ se } x_{p(j)} = 0$$

Por exemplo, $\pi = [-3, 2, -7, -4]$ representa $x_3 = 0, x_2 = 1, x_7 = 0, x_4 = 0$ com valores fixos e todos os descendentes x^q dessa solução não poderão ter os índices 3, 7 e 4 pertencendo a J_q.

Algoritmo de Balas:

Fase inicial
$\pi = \{\}$;

Fase 0 (Inicialização)
$\pi = \{\}$;
$\bar{z} = \infty$;

Fase 1
Se uma das condições de parada for verificada vá para a fase 2. No caso de ser a primeira, isto é, π está associada a uma solução viável x^p do problema (P), então neste caso se $cx^p < \bar{z}$ far-se-á $\bar{z} = cx^p$ e a melhor solução até o momento é x^p;
Caso contrário, vá para a fase 3;

Fase 2
Enquanto o último elemento da pilha for negativo, removê-lo da pilha;
Se a pilha estiver vazia vá para a fase 4;
Caso o último elemento da pilha seja positivo, troque seu sinal e vá para a fase 1;

Fase 3
Escolha uma variável x_l pelo critério aconselhado, ajunte à pilha o elemento l e vá para a fase 1;

Fase 4
Pare, solução ótima associada a $\bar{z}$;
Se $\bar{z} = +\infty$, então o problema (P) não admite solução viável;

Exemplo:
Resolva o modelo de programação linear em variáveis 0-1 a seguir pelo Método de Enumeração Implícita de Balas.

$$\begin{array}{lrrrrrrrrrrrrr}
\min & z & = & & 5x_1 & + & 7x_2 & + & 10x_3 & + & 3x_4 & + & x_5 & & \\
\text{s.a:} & & & - & x_1 & + & 3x_2 & - & 5x_3 & - & x_4 & + & 4x_5 & \leq & \text{-2} \\
& & & & 2x_1 & - & 6x_2 & + & 3x_3 & + & 2x_4 & - & 2x_5 & \leq & 0 \\
& & & & & & x_2 & - & 2x_3 & + & x_4 & + & x_5 & \leq & \text{-1} \\
& & & & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & \in & \{0,1\}
\end{array}$$

Variáveis de folga s_i:
$s_1 = -2 + x_1 - 3x_2 + 5x_3 + x_4 - 4x_5 \geq 0$
$s_2 = 0 - 2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 \geq 0$
$s_3 = -1 + 0x_1 - x_2 + 2x_3 - x_4 - x_5 \geq 0$

Iteração inicial:
$x^0 = (0\ 0\ 0\ 0\ 0)^t, J_0 = \{\}, s_1^0 = -2, s_2^0 = 0, s_3^0 = -1.\ \pi = \emptyset$. Tem-se:
1) x^0 é inviável pois $\exists s_i < 0$, no caso, tem-se $s_1^0 = -2 < 0$ e $s_3^0 = -1 < 0$
2) $\bar{z} = \infty$, pois não se conhece uma solução viável
3) As duas primeiras condições de parada, (i) e (ii), não são verificadas. Vejamos a terceira:
-2 + 1 + 0 + 5 + 1 + 0 = 5 $\geq$ 0
0 + 0 + 6 + 0 + 0 + 2 = 8 $\geq$ 0
-1 + 0 + 0 + 2 + 0 + 0 = 1 $\geq$ 0
que também não é verificada. Assim, necessitamos encontrar uma solução descendente de x^0.

Para isso, definiremos os seguintes conjuntos:
$A_0 = \{\}$, pois $\bar{z} = \infty$
$D_0 = \{2, 5\}$. Logo, $C_0 = \{1, 2, 3, 4, 5\} - (A_0 \cup D_0 \cup J_0) = \{1, 3, 4\}$, isto é, as variáveis x_1, x_3 e x_4 são as candidatas a tomarem o valor 1 (apenas uma entre elas o tomará).
Calculemos $d_j^0\ \forall j \in C_0$:
$d_1^0 = -1 - 2 - 1 = -4$; $d_3^0 = 0 - 3 + 0 = -3$; $d_4^0 = -1 - 2 - 2 = -5$ e $d_3^0 = \max\{d_1^0, d_3^0, d_4^0\}$, indicando que $l = 3$ e a variável x_3 deve assumir o valor 1.

Iteração 1:
$J_1 = J_0 \cup \{3\} = \{3\}$, $s_1^1 = 3$, $s_2^1 = -3$, $s_3^1 = 1$, e $x^1 = (0\ 0\ 1\ 0\ 0)^t$ não é viável de (P), pois $\exists s_i < 0$, no caso, $s_2^1 = -3 < 0$. $\pi = [3]$. Como $\bar{z} = \infty$ e:
3 + 1 + 0 + 1 + 0 = 5 $\geq$ 0
-3 + 0 + 6 + 0 + 2 = 5 $\geq$ 0
1 + 0 + 0 + 0 + 0 = 1 $\geq$ 0
nenhuma das condições de parada foi verificada. Passaremos, então, a procurar uma solução des-

cendente de x^1.
$A_1 = \{\}$, pois $\bar{z} = \infty$.
$D_1 = \{1, 4\}$. Logo: $C_1 = \{1, 2, 3, 4, 5\} - (\{1, 4\} \cup \{3\}) = \{2, 5\}$.
$d_2^1 = 0 + 0 + 0 = 0$, $d_5^1 = -1 - 1 + 0 = -2$. Assim sendo, $d_2^1 = \max\{d_2^1, d_5^1\} = 0$. Logo, $l = 2$ e x_2 é a nova variável a assumir valor 1.

Iteração 2:
$J_2 = J_1 \cup \{2\} = \{3, 2\}$, $s_1^2 = 0$, $s_2^2 = 3$, $s_3^2 = 0$, e $x^2 = (0\ 1\ 1\ 0\ 0)^t$ é viável de (P). $\pi = [3, 2]$. $z = cx^2 = c_2 + c_3 = 7 + 10 = 17$. Como $z = 17 < \bar{z} = \infty$, então $\bar{z}$ deve ser atualizado para $\bar{z} = 17$. A primeira regra de parada é satisfeita, indicando que devemos fazer *backtracking* a partir da solução corrente.

Iteração 3: (*backtracking*)
Com o *backtracking*, a variável x_2 assume agora o valor ZERO. Assim $\pi = [3, -2]$. $x^3 = (0\ 0\ 1\ 0\ 0)^t$ é descendente de x^1 e não é viável. $\bar{z} = 17$. As duas últimas condições de parada serão aplicadas não considerando a coluna de dados relativa ao índice 2, pois $x_2 = 0$ é fixo. A segunda condição de parada não é verificada; no entanto, a terceira é atendida:
$3 + 1 + 1 + 0 = 5 \geq 0$
$-3 + 0 + 0 + 2 = -1 < 0 \leftarrow$
$1 + 0 + 0 + 0 = 1 \geq 0$

Devemos, portanto, fazer novo *backtracking*.

Iteração 4: (*backtracking*)
Com o *backtracking*, $\pi = [-3]$. A solução $x^4 = (0\ 0\ 0\ 0\ 0)^t$ é descendente de x^0, mas com $x_3 = 0$ fixo. $\bar{z} = 17$. As duas primeiras condições de parada não são satisfeitas. Testemos a terceira:
$-2 + 1 + 0 + 1 + 0 = 0 \geq 0$
$0 + 0 + 6 + 0 + 2 = 8 \geq 0$
$-1 + 0 + 0 + 0 + 0 = -1 < 0 \leftarrow$

Como a terceira condição de parada é verificada, então devemos fazer novo *backtracking*.

Iteração Final: (*backtracking*)
Com o *backtracking*, $\pi = \emptyset$. Logo, a solução ótima é $x^\star = (0\ 1\ 1\ 0\ 0)^t$, com valor $\bar{z} = 17$.

8 Exercícios propostos

(1) Suponha a existência de cinco diferentes projetos a serem executados e seja x_j a variável binária de decisão tal que $x_j = 1$ se o projeto j for selecionado e 0, caso contrário. Considerando essa aplicação, qual o significado das seguintes restrições?

(a) $x_1 + x_2 + x_3 + x_4 + x_5 \geq 2$

(b) $x_1 + x_2 + x_3 + x_4 + x_5 \leq 2$

(c) $x_3 \leq x_1$

(d) $x_2 + x_3 + x_4 \leq x_1$

(e) $x_2 + x_3 + x_4 \geq x_1$

(2) No problema das p-medianas aparecem as duas restrições abaixo:

$$\sum_{i \in Facilidades} x_{ij} = 1 \ \ \forall j \in Clientes$$

$$x_{ij} \leq y_i \ \ \forall i \in Facilidades, \ \forall j \in Clientes$$

em que *Facilidades* é o conjunto de locais candidatos a sedir uma facilidade; *Clientes* o conjunto de clientes; y_i uma variável de decisão binária que assume valor unitário se no local i for instalada uma facilidade e 0, caso contrário; e x_{ij} uma variável de decisão binária que vale 1 se o cliente j for atendido pela facilidade instalada no local i e 0, caso contrário. Qual o significado de cada uma dessas restrições?

(3) Um editor de uma revista científica precisa designar um conjunto de artigos científicos para serem revisados em uma mesma época por um conjunto de revisores. Os artigos podem ser atribuídos aos revisores conforme a tabela abaixo, onde uma célula assume valor unitário se o revisor é considerado um especialista no tema tratado no artigo. Cada artigo deve ser analisado por, pelo menos, dois revisores. Um revisor, por sua vez, pode analisar um máximo de 3 artigos. Faça um modelo de programação matemática para designar os artigos científicos aos revisores, de forma que o número de revisores necessários seja mínimo.

	Artigo				
Revisor	1	2	3	4	5
A	1	0	1	0	0
B	1	0	0	0	1
C	0	1	1	1	1
D	0	1	1	0	0
E	1	0	1	0	0
F	1	0	1	1	0
G	0	1	1	0	1
H	1	0	0	1	0

(4) No início de cada período letivo, toda instituição de ensino tem que resolver o seguinte problema: Alocar as turmas de disciplinas ao conjunto de salas existentes. Em muitas instituições, em geral as particulares, as aulas são divididas em módulos de dois horários seguidos; por exemplo, das 8 às 10 horas é um módulo e das 10 às 12 horas é outro módulo. Uma estratégia de solução largamente usada para problemas de alocação satisfazendo a esta condição, consiste em para cada módulo, resolver um problema de designação (Também chamado de problema de atribuição). Para exemplificar o problema de designação envolvido, considere um conjunto T de turmas e um conjunto S de salas, como o exemplificado nas tabelas a seguir, que se referem à necessidade de salas para as turmas em um dado módulo. O problema de designação consiste, então, em alocar as turmas às salas, satisfazendo às restrições de que cada turma deve ser alocada a uma única sala e que em cada sala deve haver apenas uma única turma. Fazer um modelo de programação matemática para alocar as turmas às salas em um dado horário. Considere como função objetivo minimizar a folga na sala, sendo esta dada pela função custo c_{ij} dada pela tabela abaixo. Nesta função, quando não há folga na sala j para alocar a turma i, é adicionada uma penalidade de valor 1000. Por exemplo, ao se alocar a turma 2 na sala 4, a alocação é inviável e, assim, a função custo recebe o valor 990, correspondente à operação (30 - 40) + 1000. Observe, assim, que neste modelo, as alocações são sempre possíveis, ainda que inviáveis.

Sala	1	2	3	4	5	6
Capacidade	70	60	45	30	48	50

Turma	1	2	3	4	5
Demanda	28	40	59	63	51

	Salas					
Turmas	1	2	3	4	5	6
1	42	32	17	2	20	22
2	30	20	5	990	992	10
3	11	1	986	971	989	991
4	7	997	982	967	985	987
5	19	9	994	979	997	999

(5) A Laminação a Frio Ltda. produz bobinas, rolos e fitas de aço para estamparia. A empresa produz bobinas de 1,20 metros de largura e com diversas espessuras, que são armazenadas no estoque. Quando seus clientes fazem pedidos, as bobinas são retiradas do estoque e cortadas nas dimensões solicitadas. Conhecendo os pedidos e usando sua experiência, a área de planejamento estabelece os chamados "padrões de corte". Um padrão de corte estabelece como uma bobina deve ser cortada. Nesse caso específico, os padrões de corte definidos pela área de planejamento são cinco, conforme tabela a seguir.

Largura dos rolos demandados [cm]	Demanda	Padrões de corte				
		A	B	C	D	E
19	10	1	0	0	3	1
36	12	1	0	2	1	0
62	15	1	1	0	0	0
25	31	0	2	0	1	2
48	17	0	0	1	0	1

Por exemplo, o padrão de corte A estabelece que para cada bobina de 120 cm, rolos de 19, 36 e 62 cm devem ser criados.

No processo de corte há duas grandes fontes de custos. A primeira é referente às perdas com os cortes e a segunda, à sobra de rolos. A demanda frequentemente faz com que sobrem rolos, os quais devem ser armazenados no estoque para uso futuro. O custo da perda é de \$1,00/cm de rolo perdido. O custo da sobra é de \$0,20/cm para o rolo destinado ao estoque. Pede-se formular um modelo de programação matemática que minimize os custos com as perdas com o corte e com as sobras de rolos.

(6) Uma das práticas recentes de desvio de verbas públicas tem sido o superfaturamento de atividades relacionadas à limpeza e conservação do patrimônio público. Diferentemente das obras, limpeza é de difícil auditoria e, mais importante do que isso, a investigação tem dificuldades de avaliar o serviço realizado meses atrás (as obras, por outro lado, podem ser reavaliadas décadas depois de terem sido construídas). Para melhorar a situação, a prefeitura da cidade de São Paulo elaborou uma licitação relativa à atividade de limpeza. A licitação subdividiu o município em várias regiões e, para aumentar a transparência e diminuir a corrupção, estabeleceu que cada licitante pode ganhar a licitação em duas regiões, no máximo. Por outro lado, os licitantes podem fazer suas propostas para quantas regiões desejarem. A tabela a

seguir apresenta uma amostra das regiões e licitantes. Os campos em branco indicam que os licitantes não apresentaram proposta para a região porque acharam que o pagamento máximo era menor do que valia a região. Os valores da tabela se referem a milhões de reais.

	Licitante				
Região	A	B	C	D	E
Butanta	11	12	13	13	13
Itaquera	20	21	24	22	23
Lapa	9	10		11	12
SantoAmaro	4	5		6	5
Centro	13	14	16		

Para a amostra, defina qual o custo mínimo da licitação e qual licitante deve ficar com qual região de modo que o custo da prefeitura seja mínimo.

(7) Foi feita uma licitação para a construção de 4 trechos de uma rodovia. Participaram dessa licitação as empresas A, B, C e D, cujos preços estão listados na tabela a seguir, por trecho. Considerando que cada trecho deve ser feito por uma única construtora e que cada construtora não pode participar da construção de mais de dois trechos, faça um modelo de programação matemática para que se gaste a menor quantidade possível de recursos na construção desses trechos de rodovia.

	Trecho			
Construtora	1	2	3	4
A	500	700	300	200
B	450	1000	450	250
C	650	800	500	320
D	550	950	480	280

(8) Em uma dada empresa, os funcionários trabalham cinco dias seguidos e folgam os dois seguintes. A necessidade de funcionários por dia da semana, bem como os custos de cada funcionário que inicia sua jornada em um dado dia da semana estão listados na tabela a seguir.

Dia	Dom	Seg	Ter	Qua	Qui	Sex	Sab
Número requerido	12	20	16	13	16	19	14
Custo ($)	135	100	125	160	160	160	160

O objetivo da empresa é reduzir os custos com a contratação de funcionários. Elabore um modelo de programação matemática que minimize os custos com a contratação de funcionários. Implemente este modelo em um otimizador e determine quantos funcionários contratar em cada dia da semana, bem como o custo total da contratação semanal.

(9) No problema de p-centros, o objetivo é localizar p facilidades e designar clientes a facilidades de modo a minimizar a distância máxima de clientes a facilidades. Este problema pode ser formulado como:

$$\text{minimize} \quad r \tag{8.83}$$

$$\text{sujeito a:} \quad \sum_{i \in Facilidades} d_{ij} x_{ij} \leq r \qquad \forall j \in Clientes \tag{8.84}$$

$$\sum_{i \in Facilidades} x_{ij} = 1 \qquad \forall j \in Clientes \tag{8.85}$$

$$x_{ij} \leq y_i \qquad \forall i \in Facilidades,\ \forall j \in Clientes \tag{8.86}$$

$$\sum_{i \in Facilidades} y_i = p \tag{8.87}$$

$$y_i \in \{0,1\} \qquad \forall i \in Facilidades \tag{8.88}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in Facilidades,\ \forall j \in Clientes \tag{8.89}$$

em que *Clientes* representa o conjunto de vértices representando a localização dos clientes, *Facilidades* representa o conjunto de vértices representando os locais candidatos à instalação de uma facilidade, d_{ij} é a distância da facilidade instalada no local i ao cliente localizado em j, y_i é uma variável binária que assume valor unitário se uma facilidade for instalada no local i, x_{ij} é uma variável binária que tem valor unitário se a facilidade instalada no local i atender ao cliente do local j.

Dada a tabela a seguir, onde são dadas as coordenadas cartesianas dos locais A, $\cdots$, P, isto é, (coordx, coordy), determine pelo modelo anterior, a localização ótima de $p = 2$ facilidades considerando que todos os locais são candidatos à instalação de uma facilidade.

Local	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
coordx	30	37	49	52	20	40	21	17	31	52	51	42	31	5	12	36
coordy	40	52	49	64	26	30	47	63	62	33	21	41	32	25	42	16

Mostre, também, quais os clientes atendidos por cada facilidade, bem como o menor valor que r assume.

(10) É dado um objeto retangular de largura L e comprimento suficientemente grande. A partir desse objeto, deseja-se gerar uma unidade de vários itens retangulares de dimensões (h_i, w_i), onde h_i é o comprimento e $w_i \leq L$, a largura. Para produzir esses itens faz-se um corte guilhotinado, isto é, um corte que se estende de um lado ao outro do objeto formando uma faixa. O objetivo é minimizar a soma dos comprimentos das faixas.

Para a modelagem apresentada a seguir, considera-se que (1) o primeiro item alocado em cada faixa (mais à esquerda) é o de maior altura, (2) que a primeira faixa do objeto (mais baixa) é a mais alta e (3) que os itens são ordenados em forma decrescente em relação à altura, isto é, $h_1 \geq h_2 \geq \cdots \geq h_n$.

Assim, o *Open Dimensional Problem* guilhotinado pode ser modelado por:

$$\begin{aligned}
\min \quad & \sum_{i \in Itens} h_i y_i & & & \\
& y_j + \sum_{i \in Itens \mid i<j} x_{ij} & = & 1 & \forall j \in Itens \\
& \sum_{j \in Itens \mid j>i} w_j x_{ij} & \leq & (L - w_i)\, y_i & \forall i \in Itens \\
& y_i & \in & \{0,1\} & \forall i \in Itens \\
& x_{ij} & \in & \{0,1\} & \forall i \in Itens,\ \forall j \in Itens,\ j < i
\end{aligned}$$

Neste modelo, x_{ij} é uma variável de decisão binária que assume valor unitário caso o item j esteja alocado à faixa i e y_i é uma variável binária que vale 1 se o item i inicializa a faixa i.

O primeiro conjunto de restrições garante que cada item será alocado uma única vez. O segundo conjunto de restrições assegura que o somatório da largura dos itens alocados em cada faixa não ultrapassará a largura do objeto. As demais restrições definem que as variáveis de decisão são binárias.

Considerando um objeto retangular de largura $L = 20$ e comprimento suficientemente grande, e os itens da tabela a seguir, determinar o comprimento mínimo da soma das faixas, quais itens inicializam cada faixa, bem como os itens alocados a cada faixa.

Item	A	B	C	D	E	F	G	H	I	J
altura h_i	15	14	13	11	9	7	6	4	2	1
largura w_i	9	4	10	3	5	11	10	5	8	6

(11) Há um conjunto I de agentes, cada qual com capacidade b_i para processar um conjunto J de tarefas. Cada tarefa $j \in J$ só pode ser processada por um único agente i, demandando dele a_{ij} unidades de recurso. Sabe-se que o custo de alocar o agente i à tarefa j é c_{ij}. Faça um modelo de otimização para minimizar o custo total de alocação.

Para ilustrar este problema, conhecido como Problema Generalizado de Atribuição (PGA), sejam as tabelas 1 e 2. Na Tabela 1 mostra-se a quantidade a_{ij} de recursos requeridos para um agente i executar uma tarefa j, bem como a capacidade b_i de cada agente.

Tabela 1: Quantidade de recursos demandados pelas tarefas e Capacidade dos agentes

	Tarefas					
Agente	1	2	3	4	5	Capacidade
A	9	13	17	16	13	15
B	15	12	11	18	19	30
C	11	16	14	13	12	25

A Tabela 2 mostra o custo c_{ij} de alocação de um agente i a uma tarefa j.

Tabela 2: Custo de alocação de agentes a tarefas

	Tarefas				
Agente	1	2	3	4	5
A	4	3	7	6	3
B	5	2	1	8	9
C	1	6	4	3	2

(12) Uma serralheria dispõe de barras de 10 m de comprimento que devem ser convenientemente cortadas em barras menores, nos seguintes tamanhos e quantidades: (a) 30 barras de 3 m; (b) 35 barras de 4 m; (c) 58 barras de 5 m; (d) 51 barras de 6 m; (e) 73 barras de 7 m. Pede-se o esquema de corte que minimiza a perda total, bem como o excesso de barras menores cortadas.

(13) Um analista de sistemas deseja acessar cinco diferentes arquivos espalhados em dez diferentes discos, como mostrado na Tabela 3. Nesta tabela, cada célula (i, j) com valor 1 indica que o arquivo i encontra-se no disco j. Por exemplo: no disco 2 podem ser encontrados os arquivos 1, 3 e 5. A capacidade de armazenamento de cada um dos discos, em GB, é também apresentada na tabela. Deseja-se determinar o conjunto de discos que contenham todos os arquivos, sem repetição, de sorte que a capacidade total seja a menor possível.

Tabela 3: Arquivos por disco e capacidade dos discos

	Disco									
Arquivo	1	2	3	4	5	6	7	8	9	10
1	1	1	0	1	1	0	0	1	1	0
2	1	0	1	0	0	0	0	0	0	0
3	0	1	0	0	1	0	1	0	0	1
4	0	0	1	0	0	1	0	1	0	0
5	1	1	0	1	0	1	1	0	1	1
Cap. (GB)	30	50	10	20	10	40	30	10	20	20

(14) Uma empresa precisa programar sua produção para o próximo mês. Sabe-se que ela dispõe de uma única máquina para processar todas as encomendas e que estas podem ser executadas em qualquer ordem e não há necessidade de preparar a máquina. Determine o atraso máximo, conhecendo-se o tempo de processamento de cada encomenda (p_i), em dias, e as datas de entrega (d_i) no mês, conforme tabela a seguir.

Tarefa	A	B	C	D	E	F	G
p_i	7	4	2	5	6	3	1
d_i	13	10	6	9	3	20	5

Para resolver este problema, utilize a seguinte regra válida para problemas de sequenciamento de tarefas em uma máquina: O atraso máximo pode ser obtido, de forma ótima, pela heurística EDD (*Earliest Due Date*), isto é, sequenciando as tarefas em ordem não-decrescente das datas de entrega, ou seja, processando as tarefas na sequência $d[1] \leq d[2] \leq \cdots \leq d[n]$, onde $[i]$ indica a i-ésima tarefa e $d[i]$ sua data de entrega.

(15) Uma emissora de TV pretende programar um conjunto de propagandas em vários intervalos comerciais, de diferentes durações, ao longo de um dia. Sabe-se que estão programados $m = 50$ intervalos ao longo do dia, que cada intervalo i, com $i = 1, \cdots, 50$, dura b_i segundos, que há $n = 100$ propagandas que podem ser inseridas na programação e que cada propaganda j dura a_j segundos e traz um retorno de c_{ij} unidades monetárias se for inserida no intervalo i. Considerar que não é necessário inserir todas as propagandas na grade de programação da emissora e que cada propaganda só será veiculada uma única vez. Fazer um modelo de programação matemática para a emissora de TV planejar a inserção das propagandas de forma que o retorno financeiro seja o maior possível.

(16) Simplifique o problema a seguir, justificando sucintamente.

$$\begin{aligned}
\text{maximize} \quad & 5x_1 + 3x_2 + \quad x_3 + 2x_4 && (8.90)\\
\text{sujeito a:} \quad & x_1 + 8x_2 + 4x_3 + 2x_4 \leq 6 && (8.91)\\
& x_1 + 3x_2 + \quad x_3 + 7x_4 \geq 8 && (8.92)\\
& x_j \in \{0,1\} \quad \forall\, j = 1,2,\ldots,4 && (8.93)
\end{aligned}$$

(17) Transforme o problema de programação linear em variáveis 0-1 a seguir, em outro no qual apenas uma dentre as restrições (8.95)-(8.97) esteja ativa e as demais, inativas.

$$\begin{aligned}
\text{minimize} \quad & 4x_1 + 3x_2 + \quad x_3 + 2x_4 && (8.94)\\
\text{sujeito a:} \quad & x_1 + 3x_2 + 5x_3 + 2x_4 \leq 7 && (8.95)\\
& 2x_1 + 5x_2 + \quad x_3 + 3x_4 \leq 9 && (8.96)\\
& x_1 + 2x_2 + 3x_3 + x_4 \leq 10 && (8.97)\\
& x_j \in \{0,1\} \quad \forall\, j = 1,2,\ldots,4 && (8.98)
\end{aligned}$$

(18) Transforme o problema não-linear 0-1 seguinte em um problema de programação linear inteira mista (PLIM):

$$\begin{aligned}
\text{minimize} \quad & 4x_1x_2x_4 && (8.99)\\
\text{sujeito a:} \quad & 2x_1x_3 + x_2 + 3x_4 \leq 4 && (8.100)\\
& x_j \in \{0,1\} \quad \forall\, j = 1,2,3,4 && (8.101)
\end{aligned}$$

(19) Resolva pelo método *branch-and-bound*, com as regras (a) $\cdots$ (d), o PLI a seguir. Faça a árvore de busca e enumere a sequência de busca.

$$\begin{aligned}
\max \quad & 4x_1 + 5x_2 + 4x_3\\
\text{sujeito a:} \quad & x_1 + x_2 \leq 4\\
& x_2 + 2x_3 \leq 5\\
& 2x_1 + 2x_2 + 4x_3 \leq 15\\
& x_j \in \mathbb{Z}^+ \quad \forall j = 1,2,3
\end{aligned}$$

(a) Utilize a variante de Dank para escolher a variável a ramificar. Em caso de empate nesta regra, ramifique a variável de menor índice;

(b) Faça busca em profundidade;

(c) Escolhida a variável a ramificar, ramifique primeiro o valor menor da variável;

Referências

[1] M. Arenales, V. Armentano, R. Morabito, and H. Yanasse. *Pesquisa Operacional para cursos de Engenharia*. Editora Campus, Rio de Janeiro, 2007.

[2] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[3] Marco Cesar Goldbarg and Henrique Pacca L. Luna. *Otimização Combinatória e Programação Linear: modelos e algoritmos*. Editora Campus, 2ª edição, Rio de Janeiro, 2005.

[4] G. Lachtermacher. *Pesquisa Operacional na tomada de decisões*. Editora Campus, 2ª edição, Rio de Janeiro, 2004.

[5] N. Maculan and M. H. Costa Fampa. *Otimização Linear*. Editora da Universidade de Brasília, Brasília, 2006.

[6] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., New York, 1998.

[7] M. J. F. Souza. *Programação de Horários em Escolas: uma aproximação por metaheurísticas*. Tese de doutorado, Programa de Engenharia de Sistemas e Computação, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2000. Disponível em www.decom.ufop.br/prof/marcone/Publicacoes/tesemarcone.ps.

[8] Hamdy A. Taha. *Pesquisa Operacional*. Editora Pearson, 8ª edição, São Paulo, 2008.