



MANUAL DO LINGO

COM EXERCÍCIOS RESOLVIDOS

DE PROGRAMAÇÃO MATEMÁTICA

Marcone Jamilson Freitas Souza
Alexandre Xavier Martins
Tatiana Alves Costa
Frederico Augusto C. Guimarães
José Maria do Carmo Bento Alves
Túlio Ângelo Machado Toffolo

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

Sumário

I OTIMIZADOR LINGO	5
1 Introdução	5
2 Introdução à Linguagem de Modelagem do LINGO	5
2.1 Função Objetivo	5
2.2 Restrições	5
3 Adicionando Características à Linguagem de Modelagem	6
3.1 Rotulando as Restrições	6
3.2 Dando título ao modelo	6
3.3 Inserindo comentários	6
4 Operadores e Funções do LINGO	7
4.1 Operadores Aritméticos	7
4.2 Operadores Lógicos	7
4.3 Operadores Relacionais	8
4.4 Nível de prioridade dos operadores	8
4.5 Funções matemáticas	8
4.6 Funções de probabilidade	9
4.7 Funções de domínio	9
5 O modelo LINGO do Problema de Transporte	9
6 SETS (Conjuntos)	10
6.1 O que são SETS?	10
6.2 Seção SETS	10
6.3 Definindo Conjuntos Primitivos	11
6.4 Definindo conjuntos derivados	12
6.5 Funções sobre conjuntos	14
6.6 Funções de manipulação de conjuntos	14
7 Seção DATA	16
7.1 Introdução à seção DATA	16
7.2 Parâmetros	17
7.3 Análise “E se...”	18
7.4 Inicializando um atributo com um simples valor	18
7.5 Omitindo valores na seção DATA	19
8 Utilizando Arquivos-texto	19
8.1 Importando dados de um arquivo texto com @FILE	19
8.2 Exportando dados para um arquivo com @TEXT	21
9 Utilizando planilhas do EXCEL	22
9.1 Função @OLE	22
9.2 Importando dados do EXCEL com @OLE	22
9.3 Definindo nomes no EXCEL	24
9.4 Excluindo um nome definido no EXCEL	26

<i>Modelagem de PPL's</i>	3
9.5 Exportando dados para EXCEL com @OLE	26
9.6 Algumas considerações sobre @OLE	28
10 Embutindo planilhas do EXCEL no LINGO	29
11 Embutindo Modelos LINGO no EXCEL	31
12 Utilizando links OLE automatizados no EXCEL	33
12.1 Comando SET	37
 II Modelagem de Problemas de Programação Linear	38
 Carteira de Investimentos	38
Problema 1	38
 Mistura de Minérios com Custos	40
Problema 2	40
Problema 3	44
 Mistura de Minérios com Metas	46
Problema 4	46
 Problema das Usinas	50
Problema 5	50
 Planejamento da Produção	53
Problema 6	53
Problema 7	55
 Alocação de Pessoal	58
Problema 8	58
Problema 9	60
 Formação de Ligas	62
Problema 10	62
 Corte de Estoque Unidimensional (<i>Cutting Stock Problem</i>)	63
Problema 11	63
Problema 12	66
Problema 13	68
 Corte de Estoque Bidimensional	70
Problema 14	70
 Mochila 0-1	70
Problema 15	70
 Mochila 0-1 Múltipla	72
Problema 16	72

Mochila Inteira	74
Problema 17	74
Mochila Inteira Múltipla	76
Problema 18	76
Localização	78
Problema 19	78
Caixeiro Viajante	79
Problema 20	79
Roteamento de Veículos	81
Problema 21	81

Parte I

OTIMIZADOR LINGO

1 Introdução

O LINGO é um *software* de modelagem e resolução de problemas lineares e não-lineares de otimização.

2 Introdução à Linguagem de Modelagem do LINGO

2.1 Função Objetivo

A linguagem de modelagem do LINGO permite representar a função objetivo de forma bastante simples e intuitiva.

Para exemplificar, considere dois conjuntos, *fabricas* e *armazens*, e uma matriz *rotas* de elementos (i, j) , com $i \in \text{fabricas}$ e $j \in \text{armazens}$. As funções objetivo a seguir:

$$\begin{aligned} &\text{minimizar} \sum_{i \in \text{fabricas}} \sum_{j \in \text{armazens}} (\text{custo}_{ij} * \text{qtdEnviada}_{ij}) \\ &\text{maximizar} \sum_{i \in \text{fabricas}} \sum_{j \in \text{armazens}} (\text{lucro}_{ij} * \text{qtdEnviada}_{ij}) \end{aligned}$$

são assim representadas no LINGO:

$$\begin{aligned} \text{MIN} &= @SUM(\text{fabricas}(i): @SUM(\text{armazens}(j): \text{custo}(i,j) \times \text{qtdEnviada}(i,j))); \\ \text{MAX} &= @SUM(\text{fabricas}(i): @SUM(\text{armazens}(j): \text{lucro}(i,j) \times \text{qtdEnviada}(i,j))); \end{aligned}$$

ou, equivalentemente:

$$\begin{aligned} \text{MIN} &= @SUM(\text{rotas}(i,j): \text{custo}(i,j) * \text{qtdEnviada}(i,j)); \\ \text{MAX} &= @SUM(\text{rotas}(i,j): \text{lucro}(i,j) * \text{qtdEnviada}(i,j)); \end{aligned}$$

2.2 Restrições

Assim como na função objetivo, podemos usar a linguagem de modelagem do LINGO para representar as restrições do problema de forma simples e direta.

Para as seguintes restrições em notação matemática:

$$\begin{aligned} \sum_{j \in \text{armazens}} (\text{qtdEnviada}_{ij}) &\leq \text{capacidade}_i && \forall i \in \text{fabricas} \\ \sum_{i \in \text{fabricas}} (\text{qtdEnviada}_{ij}) &= \text{demanda}_j && \forall j \in \text{armazens} \end{aligned}$$

temos a seguinte representação no LINGO:

$$\begin{aligned} @FOR(\text{fabricas}(i): @SUM(\text{armazens}(j): \text{qtdEnviada}(i,j)) \leq \text{capacidade}(i)); \\ @FOR(\text{armazens}(j): @SUM(\text{fabricas}(i): \text{qtdEnviada}(i,j)) = \text{demanda}(j)); \end{aligned}$$

Obs: Toda restrição do tipo \forall é escrita no LINGO começando com a função @FOR.

3 Adicionando Características à Linguagem de Modelagem

3.1 Rotulando as Restrições

O LINGO permite que as restrições sejam rotuladas com nomes. Este recurso é útil por duas razões:

- Os nomes são usados nos relatórios das soluções geradas para torná-los mais fáceis de serem interpretados.
- Muitas mensagens de erro do LINGO referem-se aos nomes dados às restrições. Caso as restrições não sejam rotuladas, localizar a fonte desses erros pode ser bastante difícil.

Rotular restrições é extremamente simples. Basta inserir um nome entre colchetes no início da restrição. A formação de nomes obedece às seguintes regras:

1. Devem começar com um caracter alfabético (A-Z), sendo os subsequentes alfabéticos, numéricos (0-9), ou “underscore” (_);
2. Não podem ter mais que 32 caracteres.

Abaixo são mostrados alguns exemplos:

```
[FO] MIN = @SUM(rotas(i,j): custo(i,j) × qtdEnviada(i,j));
@FOR(armazens(j):
    [Dem_Armazem] @SUM(fabricas(i): qtdEnviada(i,j)) = demanda(j));
```

Nesse exemplo, FO é o nome dado à função custo. Já Dem_Armazem é o nome dado à restrição de que para todo armazém j , a soma da quantidade enviada por todas as fábricas deve ser igual à demanda do armazém. Observa-se que Dem_Armazem é um vetor de tantas posições quantos forem os armazéns.

3.2 Dando título ao modelo

Um título pode ser dado a um modelo, sendo o mesmo exibido no topo do relatório da solução gerada.

No exemplo a seguir, o modelo foi intitulado “Problema de Transporte”.

MODEL;

TITLE: Problema de Transporte;

3.3 Inserindo comentários

Um comentário pode ser definido utilizando um sinal de exclamação (!) e um ponto e vírgula (;), os quais determinam o seu inicio e fim, respectivamente.

Um exemplo de comentário é exibido a seguir:

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

4 Operadores e Funções do LINGO

Os operadores podem ser: binários, quando requererem dois argumentos - um imediatamente à esquerda do operador e outro imediatamente à direita - ou unários, quando requererem apenas um argumento.

4.1 Operadores Aritméticos

Operadores aritméticos são aqueles que trabalham com valores numéricos. O LINGO possui cinco operadores aritméticos binários, como mostrado a seguir:

Operador	Descrição
$^$	Exponenciação
*	Multiplicação
/	Divisão
+	Adição
-	Subtração

O único operador aritmético unário é o de negação (-) que, neste caso, é aplicado ao operando imediatamente à direita do sinal de negação.

4.2 Operadores Lógicos

No LINGO, os operadores lógicos são usados principalmente em expressões condicionais, definidas nas funções sobre conjuntos, para controlar quais dos seus membros serão incluídos ou excluídos na função. Eles também possuem um importante papel na construção dos membros de um conjunto. Esses recursos serão mostrados mais adiante.

Operadores lógicos retornam TRUE ou FALSE como resultado. O LINGO usa o valor 1 para representar TRUE e o valor 0 para representar FALSE.

A linguagem de modelagem do LINGO inclui nove operadores lógicos, sendo todos binários, à exceção do operador unário #NOT#.

Operador	Valor de Retorno
#NOT#	TRUE se o operando imediatamente a direita é FALSE
#EQ#	TRUE se os operandos são iguais
#NE#	TRUE se os operandos não são iguais
#GE#	TRUE se o operando da esquerda é maior ou igual ao operando da direita
#GT#	TRUE se o operando da esquerda é estritamente maior que o operando da direita
#LE#	TRUE se o operando da esquerda é menor ou igual ao operando da direita
#LT#	TRUE se o operando da esquerda é estritamente menor que o operando da direita
#AND#	TRUE se ambos os operandos possuem valor TRUE
#OR#	FALSE se ambos os operandos possuem valor FALSE

4.3 Operadores Relacionais

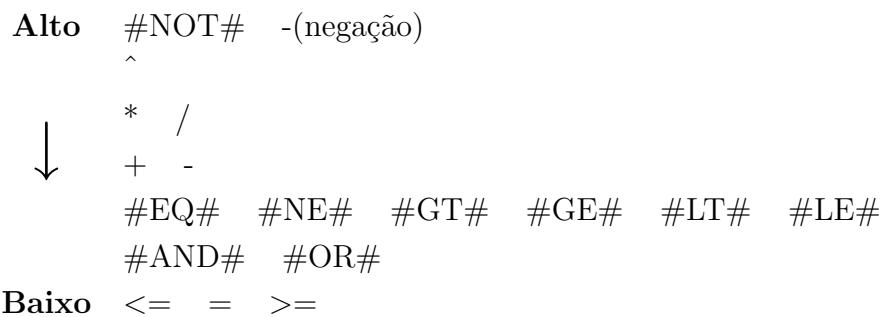
No LINGO, operadores relacionais são usados no modelo para especificar se o lado à esquerda de uma expressão deve ser igual, menor ou igual ou maior ou igual ao lado direito. Eles são usados para formar as restrições que compõem o modelo.

O LINGO faz uso de três operadores relacionais:

Operador	Descrição
=	A expressão à esquerda deve ser igual a expressão à direita
<=	A expressão à esquerda deve ser menor ou igual a expressão à direita
>=	A expressão à esquerda deve ser maior ou igual a expressão à direita

O LINGO considera o símbolo < como sendo (\leq) e, da mesma forma, (>) como sendo (\geq). Portanto, caso seja necessário usar $A < B$, a expressão deve ser convertida em $A + e \leq B$, onde e é uma constante cujo valor depende do quanto A é menor do que B . Para definir $A < 10$, com A inteiro, faça $A \leq 9$. Se A for real, faça $A \leq 10 - \epsilon$, onde $\epsilon \geq 0$ é o erro máximo admissível.

4.4 Nível de prioridade dos operadores



4.5 Funções matemáticas

O LINGO oferece um conjunto-padrão de funções matemáticas, algumas das quais são listadas a seguir:

Função	Retorno
@ABS(X)	Retorna o valor absoluto de X
@COS(X)	Retorna o cosseno de X, onde X é um ângulo em radianos
@SIN(X)	Retorna o seno de X, onde X é um ângulo em radianos
@TAN(X)	Retorna a tangente de X, onde X é um ângulo em radianos
@FLOOR(X)	Retorna o menor inteiro mais próximo de X
@SMIN(X1,X2,...,XN)	Retorna o valor mínimo dentre os elementos X1, X2, ..., e XN
@SMAX(X1,X2,...,XN)	Retorna o valor máximo dentre os elementos X1, X2, ..., e XN

4.6 Funções de probabilidade

O LINGO possui uma série de funções relacionadas à probabilidade. No entanto, apenas @RAND é apresentada aqui.

Função	Retorno
@RAND(SEED)	Retorna um número randômico entre 0 e 1, “deterministicamente” dependente da semente (SEED) definida, sendo SEED um número inteiro

4.7 Funções de domínio

Por *default*, o LINGO assume que as variáveis são contínuas e maiores ou iguais a 0. As funções de domínio impõem restrições adicionais às variáveis, determinando quais valores elas podem assumir. Essas funções e seus efeitos são descritos a seguir:

Função	Descrição
@BIN(variável)	Limita a variável a assumir apenas valores binários (0 ou 1).
@BND(inferior,variável,superior)	Limita a variável a assumir valores entre “inferior” e “superior” ($\text{inferior} \leq \text{variável} \leq \text{superior}$).
@FREE(variável)	Permite que a variável assuma quaisquer valores (-,0,+).
@GIN(variável)	Restringe a variável a assumir apenas valores inteiros.

A função @FOR pode ser usada para aplicar as funções de domínio sobre todos os membros de um atributo. Esse recurso pode ser aplicado ao modelo “Produção das Fábricas” para indicar que a quantidade enviada de uma determinada fábrica a um armazém qualquer, deve ser um valor inteiro:

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

5 O modelo LINGO do Problema de Transporte

A função objetivo e as restrições apresentadas nas seções 2.1 e 2.2 formam o modelo linear referente ao Problema de Transporte. A descrição deste problema é apresentada a seguir:

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.

O modelo LINGO do Problema de Transporte é apresentado abaixo, em que se considera a quantidade ofertada pelas fábricas maior que a soma das demandas dos armazéns.

MODEL:

TITLE: Problema de Transporte

[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));

END

O modelo LINGO completo relativo a esse problema é apresentado à página 20. Para o modelo corrente falta a definição dos conjuntos *fabricas*, *armazens* e *rotas*, bem como a entrada de dados, o que é feito por meio das seções SETS e DATA, respectivamente, explicadas a seguir.

6 SETS (Conjuntos)

6.1 O que são SETS?

SETS (conjuntos) são grupos de objetos relacionados. Um conjunto pode ser uma lista de produtos, caminhões ou empregados, por exemplo. Cada membro de um conjunto pode ter associado a si uma ou mais características. Os valores dos atributos podem ser previamente conhecidos ou não pelo LINGO. Por exemplo, cada produto em um conjunto *produtos* pode ter um atributo listando o seu preço, assim como cada caminhão em um conjunto *caminhoes* pode ter um atributo de capacidade de carga.

O LINGO reconhece dois tipos de conjuntos: primitivo e derivado.

Um conjunto primitivo é aquele formado somente por objetos que não podem ser reduzidos posteriormente. No modelo “Problema de Transporte” mostrado na seção 5, o conjunto *fabricas*, assim como o conjunto *armazens*, são exemplos de conjuntos primitivos.

Um conjunto derivado é formado por um ou mais diferentes conjuntos. O conceito chave é que ele deriva seus membros de um outro conjunto pré-existente. Novamente considerando o modelo “Problema de Transporte”, o conjunto *rotas* formado pelos pares (i, j) , onde $i \in \text{fabricas}$ e $j \in \text{armazens}$, é um conjunto derivado, pois é gerado a partir dos conjuntos primitivos *fabricas* e *armazens*. Um conjunto derivado pode também ser composto por outros conjuntos derivados.

6.2 Seção SETS

Conjuntos são definidos em uma seção opcional do modelo LINGO, chamada de seção SETS (seção de conjuntos). Antes de usar um conjunto em um modelo LINGO, é necessário listá-lo nesta seção. A seção de conjuntos é iniciada com a palavra-chave SETS: (incluindo os dois pontos), e termina com a palavra chave ENDSETS. Um modelo pode não ter uma seção SETS ou então, ter uma única ou múltiplas seções SETS. A única limitação com relação ao uso dessas seções é a necessidade de definir um conjunto e seus atributos antes de eles serem referenciados nas restrições do modelo.

6.3 Definindo Conjuntos Primitivos

Para definir um conjunto primitivo em uma seção SETS, devem ser especificados:

- o nome do conjunto;
- opcionalmente, seus membros (objetos contidos no conjunto); e
- opcionalmente, qualquer atributo que os membros dos conjuntos devam ter.

A definição de um conjunto primitivo tem a seguinte sintaxe:

```
nome [/lista_de_membros/] [: lista_de_atributos];
```

O uso dos colchetes indica que os itens ‘lista de membros’ e ‘lista de atributos’ são ambos opcionais.

O nome escolhido designa o conjunto e deve seguir as convenções padrões do LINGO. Não há distinção entre caracteres maiúsculos e minúsculos nos nomes.

Uma lista de membros constitui os membros pertencentes ao conjunto. Se os membros são incluídos na definição do conjunto, eles podem ser listados explicitamente ou implicitamente. Caso os membros não sejam incluídos na definição do conjunto, eles devem ser definidos subsequentemente na seção DATA do modelo. A seção DATA será descrita mais adiante neste capítulo.

Quando listados explicitamente, cada membro deve ter seu nome definido, opcionalmente separado por vírgulas. Os nomes dos membros devem respeitar as convenções do LINGO para nomes. O exemplo a seguir exibe os membros do conjunto *fabricas* sendo explicitamente declarados:

```
fabricas /F1 F2 F3 F4/: capacidade;
ou
fabricas /F1, F2, F3, F4/: capacidade;
```

Quando listados implicitamente, não há a necessidade de declarar o nome de cada membro. O LINGO automaticamente gera os nomes de todos os membros do conjunto. A seguir são listadas as diversas maneiras de se definir elementos de forma implícita:

```
! membros: 1, 2, 3, 4, 5;
fabricas / 1..5 /: capacidade;

! membros: cam3, cam4, cam5, cam6, cam7;
caminhoes / cam3..cam7 /: capCarga;

! membros: mon, tue, wed, thu, fri;
dias / mon..fri /: ;

! membros: oct, nov, dec, jan;
meses / oct..jan /: ;

! membros: oct2005, nov2005, dec2005, jan2006;
meses_ano / oct2005..jan2006 /: ;
```

Outra alternativa é utilizar a seção DATA como exemplificado a seguir:

```

DATA:
  n = 6;
ENDDATA

SETS:
  fabricas / 1..n /: capacidade;
ENDSETS

```

Os membros dos conjuntos podem ter um ou mais atributos especificados através da lista de atributos. Um atributo é uma propriedade definida para cada membro do conjunto. Os nomes dados aos atributos devem seguir as convenções impostas pelo LINGO. O exemplo seguinte declara dois atributos, *capacidades* e *localizacao*, para o conjunto *fabricas*.

```
  fabricas / 1..6 /: capacidade, localizacao;
```

6.4 Definindo conjuntos derivados

Para definir um conjunto derivado é necessário especificar:

- o nome do conjunto;
- seus conjuntos pais;
- opcionalmente, seus membros; e
- opcionalmente, qualquer atributo que os membros dos conjuntos devam ter.

A lista de pais é uma lista de conjuntos previamente definida, separadas por vírgulas. Caso a lista de membros não seja especificada, o LINGO constrói todas as combinações possíveis para os membros do novo conjunto derivado, utilizando os membros dos conjuntos pai. Como exemplo, considere os conjuntos abaixo:

```

SETS:
  produtos / A B /;
  maquinas / M N /;
  semana / 1..2 /;
  permitido(produtos,maquinas,semana);
ENDSETS

```

Os membros do conjunto *permitido*, exibidos a seguir, são derivados dos conjuntos *produtos*, *maquinas* e *semana*.

Índice	Membro
1	(A,M,1)
2	(A,M,2)
3	(A,N,1)
4	(A,N,2)
5	(B,M,1)
6	(B,M,2)
7	(B,N,1)
8	(B,N,2)

A lista de membros é opcional e é usada quando se deseja limitar os membros em um subconjunto de todas as combinações possíveis, derivadas dos conjuntos pais. Essa lista pode alternativamente ser especificada na seção DATA do modelo, descrita mais adiante.

Quando um conjunto não possui uma lista de membros e, portanto, contém todas possíveis combinações de membros derivados de seus conjuntos pais, ele é referido como sendo um conjunto denso. Caso ele inclua uma lista de membros que o limita em um subconjunto da sua forma densa, ele é dito conjunto esparso.

Uma lista de membros de um conjunto derivado pode ser construída usando:

- uma lista explícita de membros; ou
- um filtro de membros.

Ao declarar uma lista de membros explícitos é necessário definir todos os membros que irão pertencer ao conjunto. Cada membro listado deve ser um membro do conjunto denso formado por todas possíveis combinações dos conjuntos pais. Voltando ao conjunto *permitido*, do exemplo acima, podemos ter:

```
permitido(produtos, maquinas, semana) / A M 1, A N 2, B N 1 / ;
```

Em muitos conjuntos esparsos, todos os membros satisfazem a alguma condição que os diferencia dos não membros. Usar o filtro de membros envolve especificar uma condição lógica, a qual cada membro potencial deve satisfazer, para pertencer ao conjunto final. Uma condição lógica pode ser vista como um filtro que impede os membros, que não satisfazem algum critério, de pertencerem ao conjunto esparso. Como exemplo, suponha um conjunto de nome *caminhoes*, e que cada caminhão tenha um atributo chamado *capacidade*. Caso haja a necessidade de definir um subconjunto *carga_pesada* derivado do conjunto *caminhoes*, que contenha somente caminhões com capacidade acima de 50000 kg, teríamos:

```
carga_pesada(caminhoes) | capacidade(&1) #GT# 50000: ;
```

A barra vertical (|) marca o início do filtro de membros. O filtro permite que apenas caminhões de grande porte (com capacidade acima de 50000 kg) pertençam ao conjunto *carga_pesada*. O símbolo &1 é conhecido como um índice 'marcador de lugar' (placeholder). Ao construir um conjunto derivado que usa um filtro, o LINGO gera todas as possíveis combinações de membros originados dos conjuntos pais. Cada membro gerado é atribuído a &1 para que a condição lógica seja testada. Caso o membro passe no teste, ele é adicionado ao conjunto *carga_pesada*.

6.5 Funções sobre conjuntos

As funções @MIN e @MAX são usadas para encontrar o mínimo e o máximo de uma expressão sobre os membros de um conjunto. Considerando o modelo “Problema de Transporte” apresentado na seção 8.1, podemos ter os seguintes exemplos:

```
demanda_min = @MIN(armazens(j): demanda(j));
demanda_max = @MAX(armazens(j): demanda(j));

demanda1 = @MIN(armazens(j) | j #LE# 3: demanda(j));
demanda2 = @MAX(armazens(j) | capacidade(j) #GE# 500: demanda(j));
```

A função @SUM é utilizada para percorrer um conjunto e retornar o somatório dos valores de um determinado atributo, referentes aos membros especificados. Utilizando o modelo “Problema de Transporte”, temos:

```
demandas_total = @SUM(armazens: demanda);
ou
demanda_total = @SUM(armazens(j): demanda(j));

demanda1 = @SUM(armazens(j) | j #NE# 1: demanda(j));
demanda2 = @SUM(armazens(j) | capacidade(j) #GT# 100: demanda(j));
```

A função @FOR é usada para gerar restrições utilizando os membros de um conjunto. Ela permite escrever uma restrição apenas uma vez e o LINGO então trabalha gerando uma ocorrência da restrição para cada membro do conjunto. Como exemplo, considere um conjunto de pilhas de minério que devem ser retomadas por uma pá-carregadeira, para compor uma determinada produção diária. Para cada pilha retomada deseja-se saber o número de caçambadas realizadas pela pá-carregadeira:

```
@FOR(pilhas(i): numCacambadas(i) = retomado(i) / capCacamba);
ou, mais corretamente, lembrando que capCacamba não pode ser nulo:
@FOR(pilhas(i) | capCacamba #NE# 0: numCacambadas(i) = retomado(i) / capCacamba);
```

As funções sobre conjuntos podem ser aninhadas. As funções @SUM, @MAX e @MIN podem ser aninhadas dentro de qualquer outra função. Por outro lado, a função @FOR só pode ser aninhada dentro de outras funções @FOR. Para o modelo “Problema de Transporte”, tem-se:

```
@FOR(armazens(j): @SUM(fabricas(i): qtdEnviada(i,j)) = demanda(j));
```

6.6 Funções de manipulação de conjuntos

O LINGO oferece várias funções que ajudam a manipular os conjuntos. Elas serão descritas a seguir.

A função @IN retorna TRUE se o membro de um conjunto primitivo, especificado através de seu índice, pertence a um determinado conjunto.

```
@IN(nome_do_conjunto, indice_primitivo_1, [indice_primitivo_2 ...])
```

Como mostra o exemplo a seguir, @IN pode ser usado juntamente com os operadores lógicos, para gerar complementos de subconjuntos.

```

SETS
  fabricas / SEATTLE, DENVER, CHICAGO, ATLANTA /: ;
  fechadas(fabricas) / DENVER /: ;
  abertas(fabricas) | #NOT# @IN(fechadas, &1): ;
ENDSETS

```

O próximo exemplo ilustra como determinar se um elemento pertence ou não a um conjunto derivado específico. Note que para obter o índice dos elementos primitivos foi necessário utilizar a função @INDEX, descrita a seguir.

```

SETS
  s1 / A B C /: ;
  s2 / X Y Z /: ;
  s3(s1,s2) / A,X A,Z B,Y C,X /: ;
ENDSETS

pertence = @IN(s3, @INDEX(s1, B), @INDEX(s2, Y));

```

A função @INDEX retorna o índice de um elemento pertencente a um conjunto primitivo.

`@INDEX([nome_do_conjunto], elemento_do_conjunto_primitivo)`

Se o nome do conjunto é omitido, o LINGO retorna o índice do elemento do primeiro conjunto primitivo encontrado, cujo nome seja igual ao especificado através do elemento_do_conjunto_primitivo. Esta função é exemplificada a seguir:

```

SETS
  mulheres /DEBBIE, SUE, ALICE/ ;
  homens /BOB, JOE, SUE, FRED/;
ENDSETS

```

Neste exemplo, @INDEX(homens, SUE) retorna o valor 3. Já @INDEX(SUE) devolve o valor 2, pois o LINGO encontra primeiro o elemento SUE do conjunto *mullheres*.

A função @WRAP permite “ligar” o último elemento de um determinado conjunto ao primeiro. Isto é, quando o último (respectivamente primeiro) membro de um conjunto é atingido, por exemplo, por uma função @FOR, usar @WRAP permitirá “ligar” o índice do conjunto ao seu primeiro (respectivamente último) membro. Este recurso é particularmente útil em listas circulares.

`@WRAP (indice, limite)`

Formalmente, @WRAP retorna j de modo que $j = indice - k \times limite$, onde k é um inteiro tal que j pertença ao intervalo $[1, limite]$. Informalmente, a função @WRAP subtrai ou soma *limite* a *indice*, até que o valor a ser retornado esteja entre 1 e *limite*.

Considerando um conjunto que contenha 3 elementos, como *homens* do exemplo anterior, podemos ter os seguintes valores para @WRAP:

$$\begin{aligned}
 @WRAP(-1, 3) &= -1 - (-1) \times 3 = 2 \\
 @WRAP(0, 3) &= 0 - (-1) \times 3 = 3 \\
 @WRAP(1, 3) &= 1 - 0 \times 3 = 1 \\
 @WRAP(2, 3) &= 2 - 0 \times 3 = 2 \\
 @WRAP(3, 3) &= 3 - 0 \times 3 = 3 \\
 @WRAP(4, 3) &= 4 - 1 \times 3 = 1 \\
 @WRAP(5, 3) &= 5 - 1 \times 3 = 2
 \end{aligned}$$

A função @SIZE retorna o número de elementos contidos em um determinado conjunto, ou seja, a cardinalidade deste conjunto.

`@SIZE(nome_do_conjunto)`

O uso desta função torna o modelo mais independente, pois mesmo que o tamanho dos conjuntos se altere, o modelo ainda se manterá conciso. Isto pode ser melhor visualizado usando como exemplo a função @WRAP(indice, limite). Caso o *limite* fosse especificado como 3, qualquer alteração no tamanho do conjunto ao qual ele se refere tornaria o modelo incorreto. Por outro lado, se o *limite* fosse especificado como @SIZE(nome_do_conjunto), alterações no tamanho do conjunto não afetariam o modelo desenvolvido.

7 Seção DATA

7.1 Introdução à seção DATA

A seção DATA permite isolar os dados do resto do modelo. Isto é uma prática útil, pois facilita a manutenção do modelo e a escalabilidade das suas dimensões. Essa seção se inicia com a palavra-chave DATA: (incluindo os dois pontos) e termina com a palavra-chave ENDDATA. Na seção DATA são escritas instruções que inicializam os membros e/ou os atributos dos conjuntos, previamente instanciados na seção SETS. Essas expressões possuem a seguinte sintaxe:

`lista_de_objetos = lista_de_valores;`

A lista de objetos contém os nomes dos atributos e/ou um conjunto cujos membros serão inicializados, opcionalmente separados por vírgulas. Não pode haver mais que um nome de conjunto na lista de objetos, enquanto que vários nomes de atributo são permitidos.

A lista de valores contém os dados que serão atribuídos aos elementos da lista de objetos, opcionalmente separados por vírgulas.

A seguir são apresentadas duas maneiras de se inicializar uma lista de atributos:

```

SETS:
set1 /A,B,C/: X, Y;
ENDSETS

```

```

DATA:
X = 1, 2, 3;
Y = 4, 5, 6;
ENDDATA

```

ou

```

SETS:
set1 /A,B,C/: X, Y;
ENDSETS

```

```

DATA:
X, Y = 1, 4,
      2, 5,
      3, 6;
ENDDATA

```

Como mencionado anteriormente, membros de um conjunto podem ser inicializados na seção DATA. Utilizando esta técnica para modificar o exemplo acima, temos:

```

SETS:
set1: X, Y;
ENDSETS

DATA:
set1, X, Y = A 1 4
            B 2 5
            C 3 6;
ENDDATA

```

7.2 Parâmetros

O LINGO não restringe o uso de apenas atributos e conjuntos no lado esquerdo das instruções declaradas na seção DATA. Variáveis escalares (simples) também podem ser inicializadas na seção DATA. Quando isto ocorre, essas variáveis são referidas como parâmetros.

Como exemplo, suponha que um modelo utilize uma taxa de juros de 8.5% como um parâmetro. Isto poderia ser expresso da seguinte forma:

```

DATA:
taxa_de_juros = .085;
ENDDATA

```

Assim como os atributos, vários parâmetros podem ser inicializados em uma única instrução. Imagine agora que um novo parâmetro, *taxa_de_inflacao*, seja adicionado ao modelo acima. Deste modo, teremos:

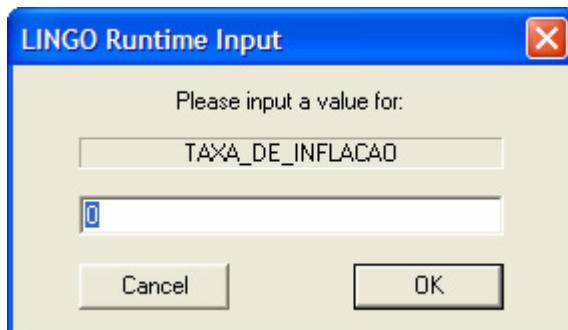
```
DATA:  
    taxa_de_juros, taxa_de_inflacao = .085, .03;  
ENDDATA
```

7.3 Análise “E se...”

Suponha que um modelo utilize a taxa de inflação como um parâmetro. Não sendo possível determinar um valor para esse parâmetro no futuro, mas sabendo que ele pode cair dentro de uma faixa entre 2% a 6%, este modelo poderia ser resolvido para vários valores de taxa de inflação, variando dentro da faixa descrita, para analisar a sensibilidade deste modelo com relação ao parâmetro. Esta análise é referida como ‘e se...’. Para definir um parâmetro deste tipo, utilize o sinal de interrogação como mostra o exemplo a seguir:

```
DATA:  
    taxa_de_inflacao = ?;  
ENDDATA
```

O LINGO exibirá uma caixa de entrada cada vez que o modelo for resolvido.



O valor digitado na caixa de entrada será atribuído ao parâmetro *taxa_de_inflacao*.

7.4 Inicializando um atributo com um simples valor

O LINGO permite inicializar todos os elementos de um atributo usando um único valor. O exemplo a seguir, mostra como isto pode ser feito:

```
SETS:  
    dias / MO, TU, WE, TH, FR, SA, SU /: necessidade;  
ENDSETS  
  
DATA:  
    necessidade = 20;  
ENDDATA
```

Se existem múltiplos atributos no lado esquerdo da instrução, será necessário um valor no lado direito para cada atributo utilizado. Acrescentando ao exemplo um novo atributo *custo*, temos:

```

SETS:
  dias / MO, TU, WE, TH, FR, SA, SU/: necessidade, custo;
ENDSETS

DATA:
  necessidade, custo = 20, 100;
ENDDATA

```

7.5 Omitindo valores na seção DATA

Valores em uma instrução da seção DATA podem ser omitidos, caso não seja possível determiná-los para alguns membros. Como exemplo, suponha uma fábrica que deseja planejar a sua capacidade para os próximos 5 anos. Além disso, suponha que a expansão da capacidade leve tempo para ser encorajada e implementada. Sendo assim, seria impossível aumentar a capacidade nos dois primeiros anos. Neste caso, teríamos:

```

SETS:
  anos /1..5/: capacidade;
ENDSETS

DATA:
  capacidade = 34, 34, , , ;
ENDDATA

```

O LINGO estará, portanto, livre para decidir os valores da capacidade para os três últimos anos.

8 Utilizando Arquivos-texto

8.1 Importando dados de um arquivo texto com @FILE

A função de interface @FILE permite importar dados de um arquivo texto para um modelo qualquer. Isto é particularmente útil para incorporar dados, gravados em arquivos, às seções SETS e DATA. A sintaxe da função @FILE é apresentada a seguir:

```
@FILE('nome_do_arquivo');
```

Quando esta função é utilizada no modelo, o LINGO irá ler os dados do arquivo especificado até que o fim do arquivo seja atingido, ou uma marca de fim de registro (~) seja encontrada. Para subsequentes @FILE referidos em um mesmo modelo, que fazem uso de um mesmo arquivo, o LINGO retoma a leitura do arquivo do ponto onde parou. Funções @FILE não podem ser aninhadas (embutir um @FILE em um arquivo que é chamado por um @FILE).

Como exemplo, considere o modelo “Problema de Transporte” apresentado na seção 5, para 6 fábricas e 8 armazéns. Sendo assim, tem-se:

SETS:

```
fabricas / F1 F2 F3 F4 F5 F6 /: capacidade;
armazens / A1 A2 A3 A4 A5 A6 A7 A8 / : demanda;
rotas(fabricas,armazens): custo, qtdEnviada;
```

ENDSETS

DATA:

```
capacidade = 60 55 51 43 41 52;
```

```
demand = 35 37 22 32 41 32 43 38;
```

```
custo = 6 2 6 7 4 2 5 9
        4 9 5 3 8 5 8 2
        5 2 1 9 7 4 3 3
        7 6 7 3 9 2 7 1
        2 3 9 5 7 2 6 5
        5 5 2 2 8 1 4 3;
```

ENDDATA

Com o objetivo de isolar completamente os dados do modelo, estes podem ser movidos para um arquivo texto. A modificação realizada é apresentada a seguir.

SETS:

```
fabricas / @FILE( 'Transporte.ldt' ) /: capacidade;
armazens / @FILE( 'Transporte.ldt' ) / : demanda;
rotas(fabricas,armazens): custo, qtdEnviada;
```

ENDSETS

DATA:

```
capacidade = @FILE( 'Transporte.ldt' );
```

```
demand = @FILE( 'Transporte.ldt' );
```

```
custo = @FILE( 'Transporte.ldt' );
```

ENDDATA

Neste modelo, o arquivo 'Transporte.ldt' (a extensão '.ldt' é usada por convenção) dispõe os dados da seguinte maneira:

! Lista das fábricas;

F1 F2 F3 F4 F5 F6 ~

! Lista dos armazéns;

A1 A2 A3 A4 A5 A6 A7 A8 ~

! Capacidade das fábricas;

60 55 51 43 41 52 ~

! Demanda dos armazéns;

35 37 22 32 41 32 43 38 ~

! Custo de envio;

6 2 6 7 4 2 5 9

4 9 5 3 8 5 8 2

5 2 1 9 7 4 3 3

7 6 7 3 9 2 7 1

2 3 9 5 7 2 6 5

5 5 2 2 8 1 4 3

As seções do arquivo de dados entre ~ são chamadas de registros. Se um arquivo não contém nenhum sinal ~, o LINGO lerá o arquivo inteiro como um único registro.

O exemplo apresentado acima esclarece como as funções @FILE trabalham ao longo do modelo. A primeira chamada a @FILE abre o arquivo “Transporte.ldt” e lê o primeiro registro (elementos do conjunto *fabricas*). A segunda chamada diz respeito ao segundo registro (elementos do conjunto *armazens*), e assim por diante.

O ultimo registro (custos de envio) não necessita de um ~. Quando o LINGO encontra o fim do arquivo, lê o último registro e o arquivo é fechado. Caso seja incluído um ~ no final deste registro, o LINGO não fechará o arquivo até que o modelo corrente seja resolvido. Isto pode causar problemas, caso múltiplos arquivos sejam abertos durante a resolução do modelo.

Comentários inclusos nos arquivos-texto são ignorados pelo LINGO. O número máximo de arquivos que um modelo pode referenciar, simultaneamente, é 16.

8.2 Exportando dados para um arquivo com @TEXT

A função de interface @TEXT pode ser usada com o intuito de exportar soluções para um arquivo texto. Podem ser exportados tanto membros de um conjunto, quanto valores de um atributo. A sintaxe para este comando é apresentada a seguir.

```
@TEXT(['nome_do_arquivo'])
```

Instruções escritas na seção DATA, que utilizam funções de interface para exportar dados, são referidas como operações de saída. Operações deste tipo são executadas somente quando o LINGO termina de resolver o modelo, seguindo a ordem a qual elas são listadas na seção.

Um exemplo de como exportar dados utilizando @TEXT é apresentado abaixo.

DATA:

```
@TEXT('Resultados.txt ') = x;
```

```
@TEXT() = y;
```

ENDDATA

No caso em que o nome do arquivo é omitido, os dados são enviados para a tela de resultados do LINGO.

9 Utilizando planilhas do EXCEL

9.1 Função @OLE

@OLE é uma função de interface usada para mover dados entre o LINGO e o Excel, através de transferências baseadas em OLE (*Object Linking and Embedding*). Essas transferências são realizadas diretamente pela memória e, portanto, não fazem uso de arquivos intermediários.

9.2 Importando dados do EXCEL com @OLE

A função @OLE pode ser usada nas seções SETS e DATA para importar dados. @OLE pode tanto ler membros de conjuntos quanto atributos - membros são esperados no formato texto, enquanto que atributos no formato numérico.

A sintaxe da função @OLE, quando usada na seção DATA para importar dados do EXCEL, é:

```
lista_de_objetos = @OLE('nome_do_arquivo.xls' [, lista_de_nomes]);
```

A lista de objetos é formada por objetos do modelo, opcionalmente separados por vírgulas, que são inicializados com dados importados de uma planilha. Ela pode conter qualquer combinação de nomes de conjuntos, atributos e variáveis escalares.

A lista de nomes é composta por nomes de campos definidos na planilha do Excel, que compõem os dados importados. Cada nome da lista deve possuir um elemento correspondente na lista de objetos.

Existem três opções possíveis de definir a lista de nomes, exemplificadas a seguir.

```
custo, capacidade = @OLE('Transporte.xls');
```

Quando a lista de nomes é omitida, o LINGO utiliza os nomes contidos na lista de objetos. Desta forma, *custo* e *capacidade* são inicializados com os valores definidos, respectivamente, nos campos nomeados como '*custo*' e '*capacidade*' no arquivo '*Transporte.xls*'.

```
custo, capacidade = @OLE('Transporte.xls', 'tabela');
```

Neste exemplo, um único campo, rotulado com o nome de '*tabela*' no arquivo '*Transporte.xls*', é utilizado para inicializar os atributos *custo* e *capacidade*. Assumindo que o campo '*tabela*' possui duas colunas, o LINGO utiliza a primeira coluna para inicializar *custo* e a segunda para inicializar *capacidade*. Para que este método funcione é necessário que tanto *custo* quanto *necessidade* sejam conjuntos de mesmo tamanho. Além disso, ambos devem ser conjuntos ou atributos - tipos diferentes não são permitidos.

```
custo, capacidade = @OLE('Transporte.xls', 'cust','cap');
```

Neste caso, cada atributo é inicializado por um campo correspondente. O atributo *custo* irá receber os dados referentes a '*cust*' e *capacidade*, os dados referentes a '*cap*'.

Para melhor entendimento de como importar os dados do EXCEL utilizando a função @OLE, o modelo “Problema de Transporte” com 6 fábricas e 8 armazens será modificado para receber os dados inseridos no arquivo '*Transporte.xls*'. As modificações são apresentadas a seguir.

SETS:

fabricas: capacidade;

armazens: demanda;

rotas(fabricas,armazens): custo, qtdEnviada;

ENDSETS

DATA:

fabricas, armazens, capacidade, demanda, custo =

@OLE('Transporte.xls', 'fabricas','armazens','capacidade','demanda', 'custo');

ENDDATA

	A	B	C	D	E	F	G	H	I	J	K	L
2												
4												
Problema de Transporte												
5	Fábricas	A1	A2	A3	A4	A5	A6	A7	A8	Capacidade		
6	F1	6	2	6	7	4	2	5	9	60		
7	F2	4	9	5	3	8	5	8	2	55		
8	F3	5	2	1	9	7	4	3	3	51		
9	F4	7	6	7	3	9	2	7	1	43		
10	F5	2	3	9	5	7	2	6	5	41		
11	F6	5	5	2	2	8	1	4	3	52		
12	Demandas	35	37	22	32	41	32	43	38			
13												

Plan1 / Plan2 / Plan3 /

Além de inserir os dados na planilha, é necessário definir os nomes para os campos 'fabricas', 'armazens', 'capacidade', 'demanda' e 'custo'. Especificamente, serão definidos os seguintes nomes:

Nome	Campo
fabricas	B6:B11
armazens	C5:J5
capacidade	K6:K11
demandas	C12:J12
custo	C6:J11

Como os objetos do modelo possuem os mesmos nomes que os seus correspondentes campos, a lista de nomes pode ser omitida na função @OLE, gerando uma nova versão simplificada para o modelo:

DATA:

fabricas, armazens, capacidade, demanda, custo = @OLE('Transporte.xls');

ENDDATA

De forma alternativa, @OLE ainda pode ser usada na seção SETS para importar membros de um conjunto. Sendo assim, para o exemplo em questão, teríamos:

SETS:

fabricas / @OLE('Transporte.xls', 'fabricas') /: capacidade;

armazens / @OLE('Transporte.xls', 'armazens') /: demanda;

rotas(fabricas,armazens): custo, qtdEnviada;

ENDSETS

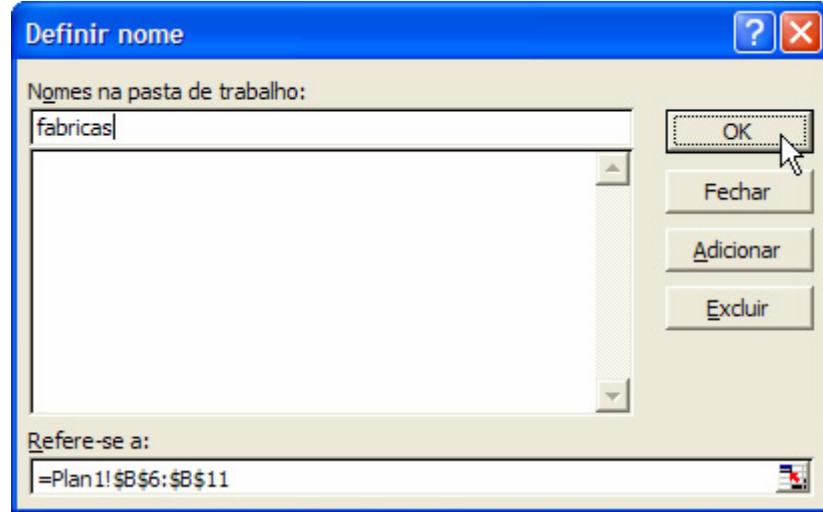
9.3 Definindo nomes no EXCEL

Existem duas maneiras de definir um nome para um campo no EXCEL.

Para definir um nome pela primeira maneira, siga os seguintes passos:

1. pressionando o botão esquerdo do mouse, arraste o cursor sobre o conjunto de células que compõem o campo;
2. solte o botão esquerdo do mouse;
3. selecione o menu Inserir|Nome|Definir;
4. digite o nome desejado; e
5. clique no botão OK.

Fábricas	Armazéns								Capacidade
	A1	A2	A3	A4	A5	A6	A7	A8	
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
Demandas	35	37	22	32	41	32	43	38	



Pela segunda maneira, mais simples, os passos são os seguintes:

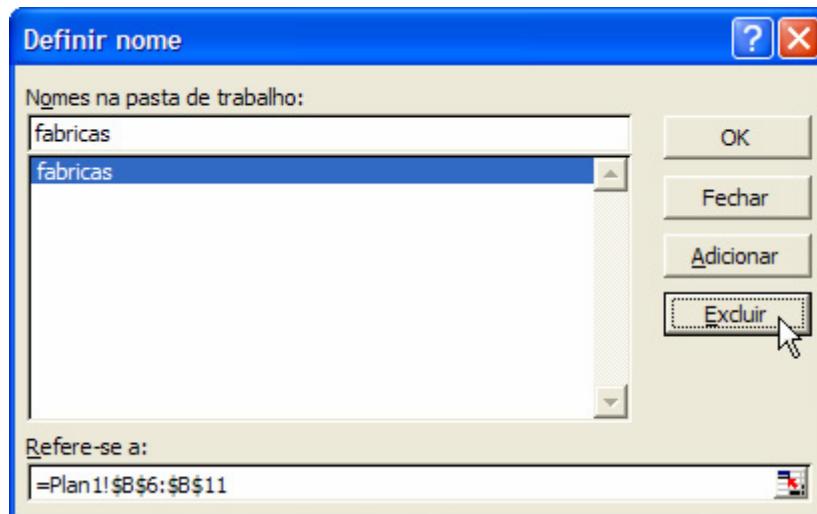
1. pressionando o botão esquerdo do mouse, arraste o cursor sobre o conjunto de células que compõem o campo;
2. solte o botão esquerdo do mouse;
3. clique na caixa de nomes localizada no canto superior esquerdo do EXCEL;
4. digite o nome desejado; e
5. pressione ENTER.

Fabricas	Armazéns								Capacidade
	A1	A2	A3	A4	A5	A6	A7	A8	
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
Demand	35	37	22	32	41	32	43	38	

9.4 Excluindo um nome definido no EXCEL

Para excluir um nome anteriormente definido no EXCEL, siga os seguintes passos:

1. selecione o menu Inserir|Nome|Definir;
2. selecione o nome desejado;
3. clique no botão Excluir; e
4. clique no botão Fechar.



9.5 Exportando dados para EXCEL com @OLE

A função de interface @OLE também pode ser usada na seção DATA com o intuito de exportar dados para o EXCEL.

A sintaxe da função @OLE, quando utilizada para exportar dados, é:

```
@OLE('nome_do_arquivo.xls' [, lista_de_nomes]) = lista_de_objetos;
```

A lista de objetos é uma lista de nomes de conjuntos, atributos e variáveis escalares, separados por vírgulas.

A lista de nomes é composta por nomes de campos definidos na planilha do Excel, que compõem as células para as quais os dados serão exportados. Cada nome da lista deve possuir um elemento correspondente na lista de objetos.

Existem três opções disponíveis de definir a lista de nomes, exemplificadas a seguir.

```
@OLE('Transporte.xls') = x, fo;
```

Quando a lista de nomes é omitida, o LINGO utiliza os nomes contidos na lista de objetos. Deste modo, o campo 'x' irá receber os dados da variável *x* e o campo 'fo', o valor de *fo*.

```
@OLE('Transporte.xls', 'solucao') = x, y;
```

Neste exemplo, um único campo, rotulado com o nome de 'solucao' no arquivo 'Transporte.xls', é utilizado para receber os valores das variáveis *x* e *y*. Assumindo que o campo 'solucao' possui duas colunas, o LINGO utiliza a primeira para receber *x* e a segunda para receber *y*. Para que este método funcione é necessário que tanto *x* quanto *y* sejam conjuntos de mesmo tamanho. Além disso, ambos devem ser conjuntos ou atributos - tipos diferentes não são permitidos.

```
@OLE('Transporte.xls', 'qtdEnviada', 'custoTotal') = x, fo;
```

Neste caso, cada campo definido em 'Transporte.xls' recebe os dados de um objeto correspondente. O campo 'qtdEnviada' irá receber os valores da variável *x* enquanto que 'custoTotal', o valor de *fo*.

Para elucidar como a função @OLE é utilizada para enviar dados ao EXCEL, o modelo "Problema de Transporte" com 6 fábricas e 8 armazéns será modificado de modo a imprimir os resultados no arquivo 'Transporte.xls'. As modificações são apresentadas abaixo.

SETS:

```
fabricas / @OLE('Transporte.xls', 'fabricas') /: capacidade;
```

```
armazens / @OLE('Transporte.xls', 'armazens') /: demanda;
```

```
rotas(fabricas, armazens): custo, qtdEnviada;
```

ENDSETS

DATA:

```
capacidade, demanda, custo =
```

```
    @OLE('Transporte.xls', 'capacidade', demanda', 'custo');
```

ENDDATA

DATA:

```
@OLE('Transporte.xls', 'qtdEnviada', 'cTotal') = qtdEnviada, fo;
```

ENDDATA

	A	B	C	D	E	F	G	H	I	J	K
10											
17											
Solução											
Armazéns											
19	Fábricas	A1	A2	A3	A4	A5	A6	A7	A8		
20	F1										
21	F2										
22	F3										
23	F4										
24	F5										
25	F6										
26											
27											
28	Custo Total:										
29											

Os campos 'qtdEnviada' e 'cTotal' devem ser definidos no arquivo 'Transporte.xls' da seguinte maneira:

Nome	Campo
qtdEnviada	C21:J26
cTotal	I28

Quando o modelo for resolvido, o LINGO lerá os dados do arquivo Excel "Transporte.xls" para as variáveis *fabricas*, *armazens*, *capacidade*, *demand* e *custo*. Uma vez que a solução ótima tenha sido encontrada, o LINGO enviará os resultados para o mesmo arquivo Excel, através dos campos 'qtdEnviada' e 'cTotal', como mostra a figura a seguir.

9.6 Algumas considerações sobre @OLE

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

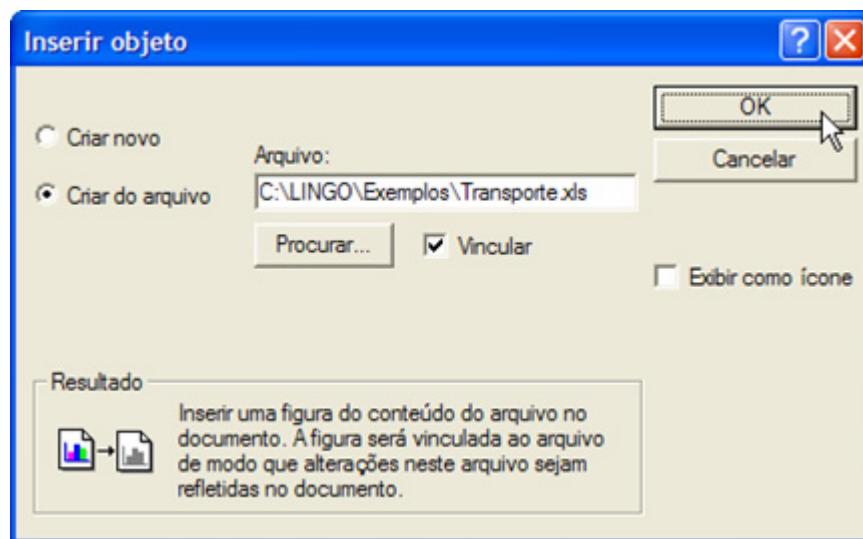
1. Para que @OLE funcione corretamente é necessário que o arquivo '.xls' utilizado esteja aberto, a não ser quando objetos embutidos são utilizados (objetos embutidos são explicados na próxima seção);
 2. A função @OLE não trabalha com conjuntos derivados tri-dimensionais; e
 3. @OLE lê os campos definidos no Excel, de acordo com a seguinte ordem: da esquerda para direita e de cima para baixo.

10 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.



Para ilustrar este processo, o modelo 'Problema de Transporte' apresentado na seção 6 será utilizado como exemplo. Após inserir o novo objeto contendo o arquivo 'Transporte.xls', temos:

LINGO - [LING01]

File Edit LINGO Window Help

Problema de Transporte

Fábricas	Armazéns								Capacidade
	A1	A2	A3	A4	A5	A6	A7	A8	
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
Demandas	35	37	22	32	41	32	43	38	

Solução

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

Custo Total:

```

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

DATA:
    capacidade, demanda, custo =
        @OLE('Transporte.xls','capacidade','demanda', 'custo');
ENDDATA

[fo] MIN = @SUM(rotas(i,j): custo(i,j)*qtdEnviada(i,j));

@FOR(fabricas(i): @SUM(armazens(j): qtdEnviada(i,j)) <= capacidade(i));
@FOR(armazens(j): @SUM(fabricas(i): qtdEnviada(i,j)) = demanda(j));

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

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

```

Ready MOD Ln 1, Col 3 5:3

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.

The screenshot shows the LINGO software window titled 'LINGO - [LINGO1]'. The main area displays two tables: 'Problema de Transporte' (Transport Problem) and 'Solução' (Solution).
Problema de Transporte:

Fábricas	Armazéns								Capacidade
	A1	A2	A3	A4	A5	A6	A7	A8	
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
Demandas	35	37	22	32	41	32	43	38	

Solução:

Fábricas	Armazéns							
	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

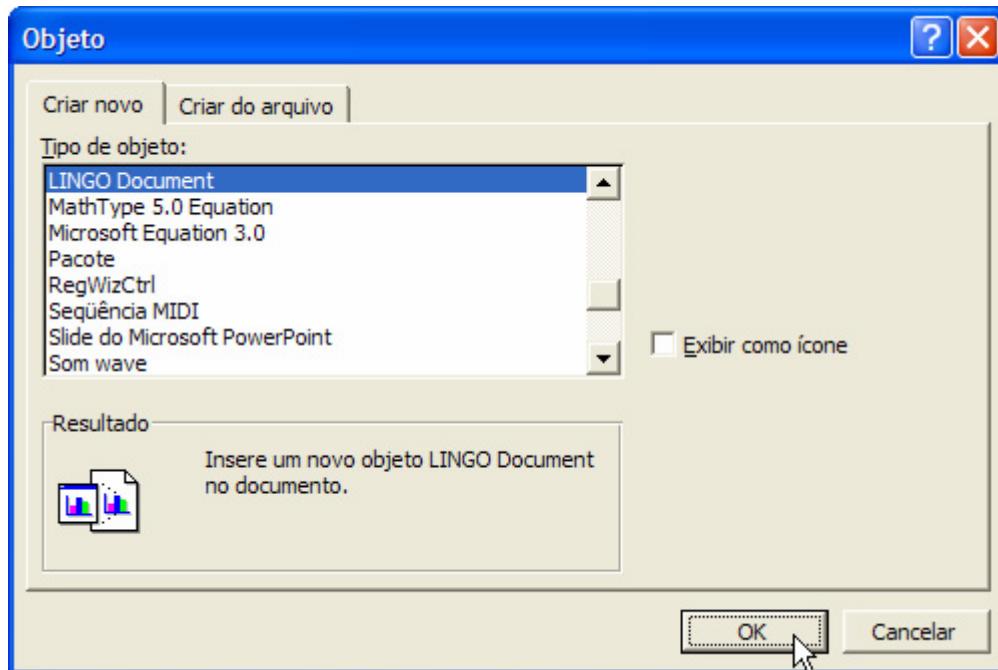
Custo Total: 664

11 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 5. Embutindo este modelo em um arquivo nomeado como ‘Transporte.xls’, teríamos:

Problema de Transporte								
Fábricas	Armazéns							
	A1	A2	A3	A4	A5	A6	A7	A8
F1	6	2	6	7	4	2	5	9
F2	4	9	5	3	8	5	8	2
F3	5	2	1	9	7	4	3	3
F4	7	6	7	3	9	2	7	1
F5	2	3	9	5	7	2	6	5
F6	5	5	2	2	8	1	4	3
Demandas	35	37	22	32	41	32	43	38

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

Custo Total:

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.

```

Microsoft Excel - LINGO em Transporte.xls
Arquivo Edit LINGO Janela Help
File Edit View Insert Cell Formulas Data Tools Window Help
A B C D E F G H I J K L M N O P Q R
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
2 \ Plan1 / Plan2 / Plan3 / For Help, press F1
  
```

Problema de Transporte

Fábricas	Armazéns								Capacidade
	A1	A2	A3	A4	A5	A6	A7	A8	
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
Demandas	35	37	22	32	41	32	43	38	

Solução

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

Custo Total: =SUM(C10:J10)

```

SETS:
  fabricas / #OLE('Transporte.xls', 'fabricas') /:
  capacidade;
  armazens / #OLE('Transporte.xls', 'armazens') /:
  demanda;

  rotas(fabricas,armazens): custo, qtdEnviada;
ENDSETS

DATA:
  capacidade, demanda, custo =
  #OLE('Transporte.xls','capacidade','demanda',
  "custo");
ENDATA

!fo1 MIN = #SUM(rotas(i,j): custo(i,j)*qtdEnviada
(i,j));

#FOR(fabricas(i): #SUM(armazens(j): qtdEnviada
(i,j)) <= capacidade(i));
#FOR(armazens(j): #SUM(fabricas(i): qtdEnviada
(i,j)) = demanda(j));

#FOR(rotas(i,j): #GIN(qtdEnviada(i,j)));
DATA:
  #OLE('Transporte.xls','solucao','cTotal') =
  qtdEnviada, fo;
ENDATA
  
```

12 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 5. Esta ilustração assume que o leitor esteja razoavelmente familiarizado com o uso de macros do Visual Basic.

Considere a seguinte planilha do EXCEL:

	A	B	C	D	E	F	G	H	I	J	K	L			
2	Problema de Transporte														
4	Fábricas	Armazéns								Capacidade					
6		A1	A2	A3	A4	A5	A6	A7	A8						
7		F1	6	2	6	7	4	2	5	9	60				
8		F2	4	9	5	3	8	5	8	2	55				
9		F3	5	2	1	9	7	4	3	3	51				
10		F4	7	6	7	3	9	2	7	1	43				
11		F5	2	3	9	5	7	2	6	5	41				
12		F6	5	5	2	2	8	1	4	3	52				
13	Demanda		35	37	22	32	41	32	43	38					
14															
15															
17	Solução														
19	Fábricas	Armazéns								Capacidade					
20		A1	A2	A3	A4	A5	A6	A7	A8						
21		F1													
22		F2													
23		F3													
24		F4													
25		F5													
26		F6													
27															
28	Custo Total:								Solve						
29															
30															
		Dados	/	Modelo	/										

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

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

Existe ainda uma segunda planilha, chamada 'Modelo', contendo o modelo do "Problema de Transporte" descrito em código *script*. Um *script* deve possuir o seguinte esquema:

SET ECHOIN 1
Outras funções SET

MODEL:
modelo LINGO
END

GO
SET ECHOIN 0
QUIT

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.

A planilha 'Modelo' é exibida abaixo:

```

1 SET ECHOIN 1
2 MODEL:
3
4 SETS:
5   fabricas / @OLE('Transporte.xls', 'fabricas') /: capacidade;
6
7   armazens / @OLE('Transporte.xls', 'armazens') /: demanda;
8
9   rotas(fabricas,armazens): custo, qtdEnviada;
10 ENDSETS
11
12 DATA:
13   capacidade, demanda, custo =
14     @OLE('Transporte.xls','capacidade','demanda', 'custo');
15 ENDDATA
16
17 [fo] MIN = @SUM(rotas(i,j): custo(i,j)*qtdEnviada(i,j));
18
19 @FOR(fabricas(i): @SUM(armazens(j): qtdEnviada(i,j)) <= capacidade(i));
20 @FOR(armazens(j): @SUM(fabricas(i): qtdEnviada(i,j)) = demanda(j));
21
22 @FOR(rotas(i,j): @GIN(qtdEnviada(i,j)));
23
24 DATA:
25   @OLE('Transporte.xls','solucao','cTotal') = qtdEnviada, fo;
26 ENDDATA
27 END
28
29 GO
30 SET ECHOIN 0
31 QUIT

```

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

Nome	Campo
modelo	A1:A31

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

```

Private Sub Solve_Click()
    Dim iErr As Integer
    Dim LINGO As Object

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

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

```

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 nenhum 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 ilustra a próxima figura.

Problema de Transporte

Fábricas	Armazéns								Capacidade
	A1	A2	A3	A4	A5	A6	A7	A8	
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
Demandas	35	37	22	32	41	32	43	38	

Solução

Fábricas	Armazéns							
	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 /

12.1 Comando SET

O comando SET permite alterar configurações padrões do LINGO. Todas as opções configuráveis pelo usuário estão disponíveis através do comando SET. A sintaxe para este comando 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 <i>status</i> do processo de busca (1: sim, 0: não)
33	ECHOIN	0	Envia comandos <i>script</i> 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	1×10^{-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 <i>probing</i> , 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 <i>r</i> 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 <i>r</i> , que indica ao método B & B para somente buscar soluções inteiras cujo valor seja pelo menos <i>r</i> unidades melhores que a melhor solução inteira encontrada até então. Acelera a busca, mas pode não garantir a solução ótima.
16	HURDLE	<i>none</i>	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.

Parte II

Modelagem de Problemas de Programação Linear

(1) **Carteira de Investimentos** A LCL Investimentos gerencia recursos de terceiros através da escolha de carteiras de investimentos para diversos clientes, baseados em *bonds* de diversas empresas. Um de seus clientes exige que:

- Não mais de 25% do total aplicado deve ser investido em um único investimento;
- Um valor superior ou igual a 50% do total aplicado deve ser investido em títulos de maturidade maiores que 10 anos;
- O total aplicado em títulos de alto risco deve ser, no máximo, de 45% do total investido.

A tabela abaixo mostra os dados dos títulos selecionados.

Título	Retorno anual	Maturidade (Anos)	Risco
1	8,7%	15	1- Muito baixo
2	9,5%	12	3- Regular
3	12,0%	8	4- Alto
4	9,0%	7	2- Baixo
5	13,0%	11	4- Alto
6	20,0%	5	5- Muito alto

Determine a estratégia ótima para o investidor de forma que a rentabilidade de sua aplicação seja máxima.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$titulos$: Conjunto de títulos;
 $retorno_j$: Retorno anual do título j (%);
 $maturidade_j$: Maturidade do título j (anos);
 $risco_j$: Risco associado ao título j .

e a seguinte variável de decisão:

x_j : Percentual a ser investido no título j .

O modelo de programação matemática para o Exercício 1 é:

$$\max \quad (1/100) \times \sum_{j \in \text{titulos}} \text{returno}_j \times x_j$$

s.a:

$$\begin{aligned} x_j &\leq 25 & \forall j \in \text{titulos} \\ \sum_{j \in \text{titulos}} x_j &\geq 50 & \forall j \in \text{titulos} \mid \text{maturidade}_j \geq 10 \\ \sum_{j \in \text{titulos}} x_j &\leq 45 & \forall j \in \text{titulos} \mid \text{risco}_j \geq 4 \\ \sum_{j \in \text{titulos}} x_j &= 100 \end{aligned}$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											
27											
28											
29											
30											
31											
32											
33											
34											
35											

INVESTIMENTOS

titulos

Título	Retorno (anos)	Maturidade (anos)	Risco
1	8,70%	15	1
2	9,50%	12	3
3	12,00%	8	4
4	9,00%	7	2
5	13,00%	11	4
6	20,00%	5	5

↑ retorno ↑ maturidade ↑ risco

SOLUÇÃO

Título	1	2	3	4	5	6
Investido (%)	0	25	0	25	25	25

↑ solucao

retmax

Retorno Máximo: **12,88%**

Title: Investimentos(R).lg4;

sets:

titulos/@ole('Investimentos(R).xls','titulos')/: retorno, maturidade, risco, x;
endsets

data:

! Importa os dados do Excel;

retorno, maturidade, risco = @ole('Investimentos(R).xls','retorno','maturidade','risco');
enddata

! Maximizar o retorno com os investimentos;

[fo] max = (1/100) * @sum(titulos(j): retorno(j) * x(j));

! Não mais do que 25 % do total aplicado deve ser investido em um único título;

@for(titulos(j): x(j) <= 25);

! Investir um valor igual ou superior a 50% do total aplicado em títulos de maturidade
superior a 10 anos;

@sum(titulos(j) | maturidade(j) #ge# 10: x(j)) >= 50;

! Total aplicado em títulos de alto risco deve ser no máximo 45% do total investido;

@sum(titulos(j) | risco(j) #ge# 4 : x(j)) <= 45;

! Total aplicado em títulos deve totalizar 100%;

@sum(titulos(j): x(j)) = 100;

data:

! Exporta resultados para o excel;

@ole('Investimentos(R).xls','solucao','retmax') = x, fo;

enddata

- (2) **Mistura de Minérios com Custos** Uma mineradora recebe uma encomenda para produzir 6000 toneladas de minério atendendo a especificação abaixo.

Elemento químico	Teor Mínimo permitido (%)	Teor Máximo permitido (%)
<i>Fe</i>	44,5	49,5
<i>Al₂O₃</i>	0,27	0,37
<i>P</i>	0,035	0,043
<i>PPC</i>	2,05	2,65
<i>He</i>	38	50

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

Pilha	Fe (%)	Al_2O_3 (%)	P (%)	PPC (%)	He (%)	Massa (ton)	Custo (\$/ton)
01	52,64	0,52	0,084	4,48	45	1500	10,50
02	39,92	0,18	0,029	0,65	97	2000	12,50
03	47,19	0,50	0,050	2,52	52	1700	12,00
04	49,36	0,22	0,039	1,74	78	1450	10,00
05	43,94	0,46	0,032	2,36	41	1250	11,50
06	48,97	0,54	0,057	4,34	90	1890	11,00
07	47,46	0,20	0,047	5,07	9	1640	10,80
08	46,52	0,32	0,039	3,51	4	1124	11,20
09	56,09	0,95	0,059	4,10	80	1990	10,40
10	46,00	0,26	0,031	2,51	21	900	12,00
11	49,09	0,22	0,040	4,20	12	1540	10,30
12	49,77	0,20	0,047	4,81	12	1630	11,90
13	53,03	0,24	0,047	4,17	1	1320	12,30
14	52,96	0,29	0,052	4,81	1	1245	11,10
15	42,09	0,17	0,031	1,38	47	1859	12,10

Qual a estratégia da mineradora para atender ao pedido de forma que o custo seja mínimo?

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $parametros$: Conjunto de parâmetros ;
 $pilhas$: Conjunto de pilhas;
 tl_j : Teor mínimo admissível para o parâmetro j no produto final (%);
 tu_j : Teor máximo admissível para o parâmetro j no produto final (%);
 Qu_i : Quantidade máxima disponível na pilha i (t);
 c_i : Custo de utilização de uma tonelada de minério da pilha i ;
 t_{ij} : Teor do parâmetro j na pilha i (%);
 p : Produção total requerida, em toneladas (t).

e a seguinte variável de decisão:

- x_i : Quantidade de minério a ser retirada da pilha i , em toneladas (t).

O modelo de programação matemática para o Exercício 2 é:

$$\min \sum_{i \in pilhas} c_i \times x_i$$

s.a:

$$\sum_{i \in pilhas} (t_{ij} - tu_j)x_i \leq 0 \quad \forall j \in parametros$$

$$\sum_{i \in pilhas} (t_{ij} - tl_j)x_i \geq 0 \quad \forall j \in parametros$$

$$\sum_{i \in pilhas} x_i = p$$

$$x_i \leq Qu_i \quad \forall i \in pilhas$$

$$x_i \in \mathbb{Z}^+ \quad \forall i \in pilhas$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2														
3														
4														
5	Produção:	6.000												
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														

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

Custo Total: R\$ 65.061,60 ← ctotal

Title: Mistura(R).lg4;

sets:

```
parametros/@ole('Mistura(R).xls','parametros')/: tl, tu;
pilhas/@ole('Mistura(R).xls','pilhas')/: c, Qu, x;
matriz(pilhas,parametros): t;
endsets
```

data:

```
! Importa os dados do Excel;
tl, tu, t, Qu, p, c =
@ole('Mistura(R).xls','tl','tu','teor','Qu','p','custo');
enddata
```

! Minimizar o custo total;

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

! 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 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 variável x(i) é inteira;

```
@for(pilhas(i): @GIN(x(i))));
```

data:

Exporta os resultados para Excel;

```
@ole('Mistura(R).xls','solucao','ctotal') = x, fo;
enddata
```

- (3) Relativamente ao problema anterior, 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 nova estratégia a ser adotada?

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

<i>parametros</i>	:	Conjunto de parâmetros ;
<i>pilhas</i>	:	Conjunto de pilhas;
<i>tl_j</i>	:	Teor mínimo admissível para o parâmetro <i>j</i> no produto final (%);
<i>tr_j</i>	:	Teor desejável para o parâmetro <i>j</i> no produto final (%);
<i>Qu_i</i>	:	Quantidade máxima disponível na pilha <i>i</i> (<i>t</i>);
<i>nunidret_i</i>	:	Número de unidades de retomada na pilha <i>i</i> ;
<i>retmin</i>	:	Retomada mínima, em toneladas (<i>t</i>);
<i>c_i</i>	:	Custo de utilização de uma tonelada de minério da pilha <i>i</i> ;
<i>t_{ij}</i>	:	Teor do parâmetro <i>j</i> na pilha <i>i</i> (%);
<i>p</i>	:	Produção total requerida, em toneladas (t).

e as seguintes variáveis de decisão:

x_i : Quantidade de minério a ser retirado da pilha *i* (*t*);

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 3 é:

$$\min \quad \sum_{i \in \text{pilhas}} (c_i \times x_i)$$

s.a:

$$\sum_{i \in \text{pilhas}} (t_{ij} - tu_j)x_i \leq 0 \quad \forall j \in \text{parametros}$$

$$\sum_{i \in \text{pilhas}} (t_{ij} - tl_j)x_i \geq 0 \quad \forall j \in \text{parametros}$$

$$x_i \leq Qu_i \quad \forall i \in \text{pilhas}$$

$$\sum_{i \in \text{pilhas}} x_i = p$$

$$nunidret_i = x_i / unidret \quad \forall i \in \text{pilhas}$$

$$y_i \geq x_i / Qu_i \quad \forall i \in \text{pilhas} \mid Qu_i \neq 0$$

$$x_i \geq retmin \times y_i \quad \forall i \in \text{pilhas}$$

$$nunidret_i \in \mathbb{Z}^+ \quad \forall i \in \text{pilhas}$$

$$y_i \in \{0, 1\} \quad \forall i \in \text{pilhas}$$

Modelo Lingo

Title: Mistura(R).lg4;

sets:

parametros/@ole('Mistura(R).xls','parametros')/: tl, tu;

pilhas/@ole('Mistura(R).xls','pilhas')/: c, Qu, x, y, nunidret;

matriz(pilhas,parametros): t;

endsets

data:

! Importa os dados do Excel;

tl, tu, t, Qu, retmin, p, c, unidret =

```
@ole('Mistura(R).xls','tl','tu','teor','Qu','retmin','p','custo','unidret');
```

enddata

! Minimizar o custo total;

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

O limite superior de especificação deve ser satisfeita para cada parâmetro.

O limite inferior de especificação deve ser satisfeito para cada parâmetro

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

! A produção total deve ser igual a p ;
 $\text{@sum}(\text{pilhas}(i): x(i)) = p;$

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

! Se for retomada qualquer quantidade na pilha i então $y(i) = 1$. Caso contrário, $y(i)$ assume valor 0;
 $\text{@for}(\text{pilhas}(i) | Qu(i) \neq 0: y(i) >= x(i)/Qu(i));$

! Se for retomar alguma pilha i a quantidade $x(i)$ a retomar deve ser superior ou igual a $retmin$;
 $\text{@for}(\text{pilhas}(i): x(i) >= retmin * y(i));$

! A variável $nunidret(i)$ deve ser inteira e $y(i)$ binária;
 $\text{@for}(\text{pilhas}(i):$
 $\text{@GIN}(\text{nunidret}(i));$
 $\text{@BIN}(y(i));$
 $);$

data:
! Exporta os resultados para o Excel;
 $\text{@ole}('Mistura(R).xls', 'solucao') = x;$
enddata

- (4) **Mistura de Minérios com Metas** 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)	Custo (\$/ton)
01	52,64	0,52	0,084	4,48	45	1500	10,50
02	39,92	0,18	0,029	0,65	97	2000	12,50
03	47,19	0,50	0,050	2,52	52	1700	12,00
04	49,36	0,22	0,039	1,74	78	1450	10,00
05	43,94	0,46	0,032	2,36	41	1250	11,50
06	48,97	0,54	0,057	4,34	90	1890	11,00
07	47,46	0,20	0,047	5,07	9	1640	10,80
08	46,52	0,32	0,039	3,51	4	1124	11,20
09	56,09	0,95	0,059	4,10	80	1990	10,40
10	46,00	0,26	0,031	2,51	21	900	12,00
11	49,09	0,22	0,040	4,20	12	1540	10,30
12	49,77	0,20	0,047	4,81	12	1630	11,90
13	53,03	0,24	0,047	4,17	1	1320	12,30
14	52,96	0,29	0,052	4,81	1	1245	11,10
15	42,09	0,17	0,031	1,38	47	1859	12,10

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

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 peso do critério.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $parametros$: Conjunto de parâmetros ;
 $pilhas$: 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 (t);
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 4 é:

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

s.a:

$$\sum_{i \in \text{pilhas}} (t_{ij} - tu_j)x_i \leq 0 \quad \forall j \in \text{parametros}$$

$$\sum_{i \in \text{pilhas}} (t_{ij} - tl_j)x_i \geq 0 \quad \forall j \in \text{parametros}$$

$$\sum_{i \in \text{pilhas}} ((t_{ij} - tr_j) \times x_i) + dtn_j - dtp_j = 0 \quad \forall j \in \text{parametros}$$

$$x_i \leq Qu_i \quad \forall i \in \text{pilhas}$$

$$\sum_{i \in \text{pilhas}} x_i = p$$

$$nunidret_i = x_i / unidret \quad \forall i \in \text{pilhas}$$

$$y_i \geq x_i / Qu_i \quad \forall i \in \text{pilhas} \mid Qu_i \neq 0$$

$$x_i \geq retmin \times y_i \quad \forall i \in \text{pilhas}$$

$$nunidret_i \in \mathbb{Z}^+ \quad \forall i \in \text{pilhas}$$

$$y_i \in \{0, 1\} \quad \forall i \in \text{pilhas}$$

Modelo Lingo

A	B	C	D	E	F	G	H	I	J	K	L	M	N
MISTURA DE MINÉRIOS COM METAS													
4	Produção:	6.000	↔p		Fe (%)	Al2O3 (%)	P (%)	PPC (%)	He (%)	↔parametros			
5					Limite Superior	49,5	0,37	0,043	2,65	50	↔tu		
6					Meta	47,0	0,32	0,040	2,35	40	↔tr		
7					Limite Inferior	44,5	0,27	0,035	2,05	38	↔tl		
8					Peso Comparação	1	100	1000	10	1			
9					Critério	5	0	100	10	1			
10					Peso Desvio da Meta	5	0	100000	100	0	↔wdt		
11					Mistura	47,0	0,29	0,040	2,35	45			
12					Desvio negativo (100t)	0,00	208,60	0,00	0,00	0,00	↔dtn		
13					Desvio positivo (100t)	0,50	0,00	0,00	0,00	31060,00	↔dtp		
14													Custo Total: R\$ 70.759,00
15													
16													
17													
18													
19					Massa (t)	Fe (%)	Al2O3 (%)	P (%)	PPC (%)	He (%)	Custos (\$/t)	A retomar (t)	
20					Pilha 1	1500	52,64	0,52	0,084	4,48	45	10,50	0,00
21					Pilha 2	2000	39,92	0,18	0,029	0,65	97	12,50	500,00
22					Pilha 3	1700	47,19	0,50	0,050	2,52	52	12,00	1480,00
23					Pilha 4	1450	49,36	0,22	0,039	1,74	78	10,00	980,00
24					Pilha 5	1250	43,94	0,46	0,032	2,36	41	11,50	0,00
25					Pilha 6	1890	48,97	0,54	0,057	4,34	90	11,00	0,00
26					Pilha 7	1640	47,46	0,20	0,047	5,07	9	10,80	0,00
27					Pilha 8	1124	46,52	0,32	0,039	3,51	4	11,20	0,00
28					Pilha 9	1990	56,09	0,95	0,059	4,10	80	10,40	0,00
29					Pilha 10	900	46,00	0,26	0,031	2,51	21	12,00	730,00
30					Pilha 11	1540	49,09	0,22	0,040	4,20	12	10,30	0,00
31					Pilha 12	1630	49,77	0,20	0,047	4,81	12	11,90	0,00
32					Pilha 13	1320	53,03	0,24	0,047	4,17	1	12,30	1190,00
33					Pilha 14	1245	52,96	0,29	0,052	4,81	1	11,10	0,00
34					Pilha 15	1859	42,09	0,17	0,031	1,38	47	12,10	1120,00
35													
36													
					↑ pilhas	↑ Qu	↑ teor	↑ teor	↑ 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
```

- (5) **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 ;
- $demandaj$: 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) ;
- 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 5 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:

$$\begin{aligned} \sum_{j \in usinas} x_{ij} &\leq cap_i && \forall i \in minas \\ \sum_{i \in minas} x_{ij} &= demandaj && \forall j \in usinas \\ y_i &\geq (\sum_{j \in usinas} x_{ij}) / cap_i && \forall i \in minas \\ x_{ij} &\geq 0 && \forall i \in minas, \forall j \in usinas \\ y_i &\in \{0, 1\} && \forall i \in minas \end{aligned}$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	USINAS													
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17	SOLUÇÃO													
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														

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

- (6) **Planejamento da Produção** A LCL Motores recebeu recentemente uma encomenda para entregar 3 modelos diferentes de motores. Cada motor necessita de um determinado número de horas de trabalho no setor de montagem e de acabamento. Para atender a encomenda, a LCL pode também terceirizar parte de sua produção. A tabela a seguir resume as informações sobre a demanda por cada modelo de motor, o tempo necessário para montar uma unidade de cada modelo, a quantidade de horas disponíveis no setor de montagem, o tempo necessário para dar acabamento a uma unidade de cada unidade, a quantidade de horas disponíveis no setor de acabamento, o custo de produção, bem como o custo de terceirização de uma unidade de cada modelo:

Modelo	1	2	3	Total
Demand	3000 unid.	2500 unid.	500 unid.	6000 unid.
Montagem	1 h/unid.	2 h/unid.	0,5 h/unid.	6000 h
Acabamento	2,5 h/unid.	1 h/unid.	4 h/unid.	10000 h
Custo de produção	\$50	\$90	\$120	-
Terceirizado	\$65	\$92	\$140	-

Qual a estratégia ótima a ser adotada pela empresa de forma a atender aos pedidos?

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$modelos$: Conjunto de modelos de motores;
 dem_i : Demanda por motores do modelo i ($unid$);
 $mont_i$: Tempo gasto na montagem de um motor do modelo i ($h/unid$);
 $acob_i$: Tempo gasto no acabamento de um motor do modelo i ($h/unid$);
 $cprod_i$: Custo de produção de um motor do modelo i ;
 $cterc_i$: Custo de terceirização de um motor do modelo i ;
 $hMont$: Tempo total disponível para a montagem dos motores (h);
 $hAcab$: Tempo total disponível para o acabamento dos motores (h);

e as seguintes variáveis de decisão:

x_i : Quantidade de motores do modelo i produzidos;
 y_i : Quantidade de motores do modelo i terceirizados.

O modelo de programação matemática para o Exercício 6 é:

$$\min \quad \sum_{i \in \text{modelos}} (cprod_i \times x_i) \quad + \quad \sum_{i \in \text{modelos}} (cterc_i \times y_i)$$

S.a:

$$x_i + y_i = dem_i \quad \forall i \in modelos$$

$$\sum_{i \in \text{modelos}} (mont_i \times x_i) \leq hMont$$

$$\sum_{i \in \text{modelos}} (acab_i \times x_i) \leq hAcb$$

$$x_i \in \mathbb{Z}^+ \quad \forall i \in \text{modelos}$$

$$y_i \in \mathbb{Z}^+ \quad \forall i \in \text{modelos}$$

Modelo Lingo

Title: Motores(R).lg4;

sets:

```
modelos/@ole('Motores(R).xls','modelos')/: dem, mont, acab, cprod, cterc, x, y;
endsets
```

data:

! Importa os dados do Excel;

dem, mont, acab, cprod, cterc, hMont, hAcab =

```
@ole('Motores(R).xls','demanda','mont','acab','cprod','cterc','hMont','hAcab');
```

enddata

! Minimiza o custo com a produção e a terceirização dos motores;

```
[fo] min = @sum(modelos(i): cprod(i)*x(i)) + @sum(modelos(i): cterc(i)*y(i));
```

**! Total de modelos produzidos e terceirizados deve ser maior ou igual à
demanda requerida;**

```
@for(modelos(i): x(i) +y(i) = dem(i));
```

**! Tempo gasto na montagem dos motores deve ser menor ou igual ao
tempo disponível para esta tarefa;**

```
@sum(modelos(i): mont(i)*x(i)) <= hMont;
```

**! Tempo gasto no acabamento dos motores dever ser menor ou igual ao tempo
disponível para esta tarefa;**

```
@sum(modelos(i): acab(i)*x(i)) <= hAcab;
```

! As variável x(i) e y(i) devem ser inteiras;

```
@for(modelos(i):
```

```
    @GIN(x(i));
```

```
    @GIN(y(i));
```

```
);
```

data:

! Exporta dados para o Excel;

```
@ole('Motores(R).xls','Qprod','Qterc','ctotal') = x, y, fo;
```

enddata

- (7) **Planejamento da Produção** 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:

Máquinas	Disponibilidades (máquina-hora/mês)
M1	80
M2	20

Mão-de-obra	Disponibilidade (homem-hora/mês)
MO1	60
MO2	40

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:

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

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

O setor comercial da empresa fornece as seguintes informações:

Produtos	Potencial de vendas (unidades/mês)	Lucro Unitário (R\$/mês)
I	70	10,00
II	60	8,00
III	40	9,00
IV	20	7,00

Deseja-se planejar a produção mensal da empresa que maximize o lucro.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $produtos$: Conjunto de produtos;
 $maquinas$: Conjunto de máquinas;
 $mObras$: Conjunto dos tipos de mão-de-obra;
 dem_j : Demanda por produtos do tipo j (*unid*);
 $lucro_j$: Lucro obtido com a venda de produtos do tipo j ;
 $dispMaq_i$: Disponibilidade da máquina i (*h/mes*);
 $dispMObra_k$: Disponibilidade de mão-de-obra do tipo k (*h/mes*);
 $hMaq_{ij}$: Tempo disponível da máquina i para produção de produtos do tipo j (*h*);
 $hMObra_{kj}$: Tempo de mão-de-obra do tipo k para produção de produtos do tipo j (*h*).

e as seguintes variáveis de decisão:

- x_j : Quantidade fabricada de produtos do tipo j (*unid*);

O modelo de programação matemática para o Exercício 7 é:

$$\max \sum_{j \in \text{produtos}} (\text{lucro}_j \times x_j)$$

s.a:

$$\sum_{j \in \text{produtos}} (hMaq_{ij} \times x_j) \leq \text{dispMaq}_i \quad \forall i \in \text{maquinas}$$

$$\sum_{j \in \text{produtos}} (hMObra_{kj} \times x_j) \leq \text{dispMObra}_k \quad \forall k \in mObra$$

$$x_j \leq \text{dem}_j \quad \forall j \in \text{produtos}$$

$$x_j \in \mathbb{Z}^+ \quad \forall j \in \text{produtos}$$

Modelo Lingo

A	B	C	D	E	F	G	H	I	J	K	L	M	N
PRODUTOS													

Title: Produtos(R).lg4;

sets:

```
produtos/@ole('Produtos(R).xls','produtos')/: demanda, lucro, x;
maquinas/@ole('Produtos(R).xls','maquinas')/: dispMaq;
mObras/@ole('Produtos(R).xls','mObras')/: dispMObra;
matriz1(maquinas,produtos): hMaq;
matriz2(mObras,produtos): hMObra;
endsets
```

data:

```
! Importa dados do Excel;
demanda, lucro, dispMaq, dispMObra, hMaq, hMObra =
@ole('Produtos(R).xls','demanda','lucro','dispMaq','dispMObra','hMaq','hMObra');
enddata
```

! Maximizar o lucro obtido com a fabricação dos diferentes tipos de produtos;
[fo] max = @sum(produtos(j): lucro(j)*x(j));

! Tempo de utilização de cada máquina deve ser \leq à disponibilidade da mesma;
@for(maquinas(i): @sum(produtos(j): hMaq(i,j)*x(j)) <= dispMaq(i));

! Mão de obra total utilizada deve ser \leq ao total de mão de obra disponível;
@for(mObras(k): @sum(produtos(j): hMObra(k,j)*x(j)) <= dispMObra(k));

! Total de produtos fabricados deve ser \leq ao demandado;
@for(produtos(j): x(j) <= demanda(j));

! A variável x(j) deve ser inteira;
@for(produtos(j): @GIN(x(j)));

data:

```
! Exportando dados para o Excel;
@ole('Produtos(R).xls','solucao','lucromax') = x, fo;
enddata
```

- (8) **Alocação de Pessoal *Staff Scheduling*** O administrador de um hospital deseja determinar o escalonamento dos enfermeiros. Para isso ele organiza um sistema de plantão dividindo o dia em 6 períodos de 4 horas. A tabela a seguir mostra o número mínimo de enfermeiros que devem estar presentes em cada horário.

Horário	8-12	12-16	16-20	20-24	24-04	04-08
# enfermeiros	51	58	62	41	32	19

Cada enfermeiro cumpre um plantão normal de 8 horas, que pode começar apenas no início de um destes períodos. No horário de 8 às 20 horas, o enfermeiro recebe R\$100 por hora de

trabalho e R\$125 por hora no horário noturno (20 horas às 18 horas). Como o administrador deve escalar os enfermeiros de forma a minimizar o custo com a mão-de-obra?

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $turnos$: Conjunto de turnos;
 req_j : Número de enfermeiros requeridos no turno j ;
 $custos_j$: Custo da jornada de trabalho no turno j ;

e a seguinte variável de decisão:

x_j : Quantidade de enfermeiros a serem escalados no início do turno j (unid);

O modelo de programação matemática para o Exercício 8 é:

$$\min \sum_{j \in turnos} (custos_j \times x_j)$$

s.a:

$$\begin{aligned} x_{j-1} + x_j &\geq req_j & \forall j \in turnos \\ x_j &\in \mathbb{Z}^+ & \forall j \in turnos \end{aligned}$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2	ENFERMEIROS													
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16	SOLUÇÃO													
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														

Turno	Horário	Enfermeiros	Custo
1	08-12	51	R\$ 800,00
2	12-16	58	R\$ 800,00
3	16-20	62	R\$ 900,00
4	20-24	41	R\$ 1.000,00
5	24-04	32	R\$ 1.000,00
6	04-08	19	R\$ 900,00

↑ turnos ↑ requerido ↑ custo

Turno	Horário	Enfermeiros	Excesso
1	08-12	51	0
2	12-16	34	27
3	16-20	28	0
4	20-24	13	0
5	24-04	19	0
6	04-08	0	0

↑ solucao ↑ excesso

Custo Total: R\$ 125.200,00 ← ctotal

Title: Enfermeiros(R).lg4;

sets:

```
turnos/@ole('Enfermeiros(R).xls','turnos')/: req, custos, x;
endsets
```

data:

! Importa dados do Excel;

```
custos, req = @ole('Enfermeiros(R).xls','custo','requerido');
enddata
```

! Minimiza o custo com pagamento de enfermeiros em cada turno;

```
[fo] min = @sum(turnos(j): custos(j)*x(j));
```

! Total de enfermeiros trabalhando em cada turno deve ser \geq ao requerido;

```
@for(turnos(j): [excesso] x(@wrap(j-1, @size(turnos))) + x(j) >= req(j));
```

! A variável x(j) deve ser inteira;

```
@for(turnos(j): @GIN(x(j))));
```

data:

! Exporta dados para o Excel;

```
@ole('Enfermeiros(R).xls','excesso','solucao','ctotal') = excesso, x, fo;
```

enddata

- (9) O administrador de um hospital deseja determinar o escalonamento dos enfermeiros. Para isso ele organiza um sistema de plantão dividindo o dia em 6 períodos de 4 horas. A tabela a seguir mostra o número mínimo de enfermeiros que devem estar presentes em cada horário.

Horário	08-12	12-16	16-20	20-24	24-04	04-08
# enfermeiros	51	58	62	41	32	19

Cada enfermeiro cumpre um plantão normal de 8 horas, que pode começar apenas no início de um destes períodos. No horário de 8 às 20 horas, o enfermeiro recebe R\$100 pela hora de trabalho e R\$125 no horário noturno (20 horas às 18 horas). Alguns enfermeiros podem ser solicitados para estender o plantão por mais 4 horas seguidas. A hora-extra custa 50% mais caro e em cada plantão, não mais que 40% dos enfermeiros podem estar cumprindo hora-extra. Como o administrador deve escalar os enfermeiros, minimizando o custo com a mão-de-obra?

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- turnos* : Conjunto de turnos;
req_j : Número de enfermeiros requeridos no turno j ;
cSHE_j : Custo da jornada de trabalho sem hora extra no turno j
cCHE_j : Custo da jornada de trabalho com hora extra no turno j

e as seguintes variáveis de decisão:

- x_j : Quantidade de enfermeiros que começam sua jornada no início do turno j e não fazem hora-extra
 y_j : Quantidade de enfermeiros que começam sua jornada no início do turno j e fazem hora-extra

O modelo de programação matemática para o Exercício 9 é:

$$\min \quad \sum_{j \in turnos} (cSHE_j \times x_j + cCHE_j \times y_j)$$

S.a:

$$x_j + y_j + x_{j-1} + y_{j-1} + y_{j-2} \geq req_j \quad \forall j \in turnos$$

$$y_{j-2} \leq 0.4 \times (x_j + y_j + x_{j-1} + y_{j-1} + y_{j-2}) \quad \forall j \in turnos$$

$$x_j \in \mathbb{Z}^+ \quad \forall j \in turnos$$

$$y_j \in \mathbb{Z}^+ \quad \forall j \in turnos$$

Modelo Lingo

Title: EnfermeirosHoraExtra(R).lg4;

sets:

```
turnos/@ole('EnfermeirosHoraExtra(R).xls','turnos')/: req, cSHE, cCHE, x, y;
endsets
```

data:

! Importa dados do Excel;

```
req, cSHE, cCHE = @ole('EnfermeirosHoraExtra(R).xls','requerido','cSHE','cCHE');
enddata
```

! Minimiza o custo com o pagamento de enfermeiros com e sem hora extra trabalhando no turno j;

```
[fo] min = @sum(turnos(j): cSHE*x(j) + cCHE*y(j));
```

@for(turnos(j):

! Total de enfermeiros em um determinado turno deve ser \geq ao requerido;

```
[excesso] x(j) + y(j) +
x(@wrap(j-1,@size(turnos))) +
y(@wrap(j-1,@size(turnos))) +
y(@wrap(j-2,@size(turnos))) >= req(j);
```

! O total de enfermeiros com hora extra não deve ultrapassar 40% do total de enfermeiros em um turno;

```
[folga] y(@wrap(j-2,@size(turnos))) <= 0.4*(x(j) + y(j) +
x(@wrap(j-1,@size(turnos))) +
y(@wrap(j-1,@size(turnos))) +
y(@wrap(j-2,@size(turnos))));
```

As variáveis $x(j)$ e $y(j)$ devem ser inteiras;

```
@GIN(x(j));
```

```
@GIN(y(j));
```

```
);
```

data:

! Importa dados para o Excel;

```
@ole('EnfermeirosHoraExtra(R).xls','excesso','folga','semHE','comHE','ctotal') =
excesso, folga, x, y, fo;
enddata
```

(10) Deseja-se formar p ligas L_s a partir de q matérias-primas R_j . Sabe-se que:

- (a) uma unidade da matéria prima R_j contém a_{ij} unidades do metal M_i ;
- (b) uma unidade da liga L_s contém b_{is} unidades do metal M_i ;
- (c) uma unidade da matéria-prima R_j custa c_j unidades monetárias;
- (d) uma unidade da liga L_s é vendida a v_s unidades monetárias;

(e) de cada matéria-prima R_j só existem u_j unidades disponíveis;

Faça um modelo de programação linear que permita determinar a quantidade x_j de matéria-prima R_j a ser comprada e a quantidade y_s de liga L_s a ser vendida para que o lucro seja máximo.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$ligas$:	Conjunto de ligas;
$metais$:	Conjunto de metais;
$materias$:	Conjunto de matérias-primas;
v_s	:	Valor de venda de uma unidade da liga s ;
p_j	:	Custo de uma unidade da matéria-prima j ;
k_j	:	Número de unidades disponíveis da matéria-prima j ;
a_{ij}	:	Número de unidades do metal i encontrado em uma unidade da matéria-prima j ;
b_{is}	:	Número de unidades do metal i presente em uma unidade da liga s ;

e as seguintes variáveis de decisão:

x_j	:	Quantidade de matéria-prima do tipo j a ser comprada
y_s	:	Quantidade de liga do tipo s a ser vendida

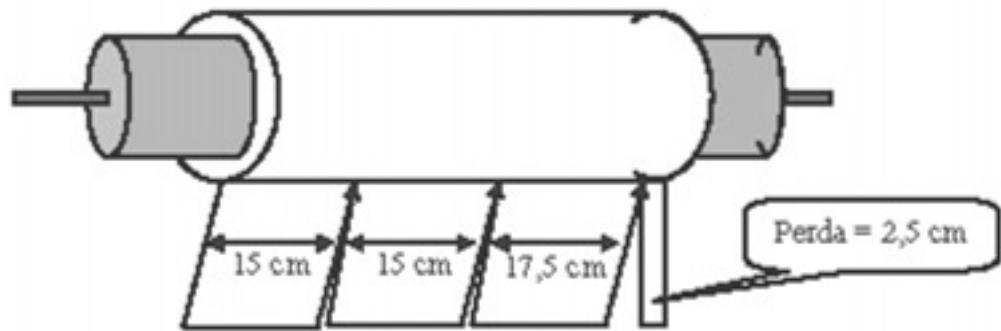
O modelo de programação matemática para o Exercício 10 é:

$$\max \sum_{s \in ligas} v_s y_s - \sum_{j \in materias} p_j x_j$$

s.a:

$$\begin{aligned} \sum_{j \in materias} a_{ij} x_j &\geq \sum_{s \in ligas} b_{is} y_s \quad \forall i \in metais \\ x_j &\leq k_j \quad \forall j \in materias \\ x_j &\in \mathbb{Z}^+ \quad \forall j \in materias \\ y_s &\in \mathbb{Z}^+ \quad \forall s \in ligas \end{aligned}$$

- (11) **Problema de Corte de Estoque Unidimensional (*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. A largura das bobinas é 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.



A tabela abaixo relaciona as possíveis programações de cortes, tendo em vista as encomendas.

Programações de corte	Largura da Faixa Cortada			Desperdício
	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

Qual 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*;
- demand_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 a seguinte variável de decisão:

- x_i* : Número de cortes realizados segundo o padrão *i*

O modelo de programação matemática para o Exercício 11 é:

$$\min \sum_{i \in padroes} (desp_i \times x_i)$$

s.a:

$$\sum_{i \in padroes} a_{ij}x_i \geq demanda_j \quad \forall j \in bobinas$$

$$\sum_{i \in padroes} a_{ij}x_i \leq demanda_j + estmax \quad \forall j \in bobinas$$

$$x_i \in \mathbb{Z}^+ \quad \forall i \in padroes$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N												
1																										
2	CORTE UNIDIMENSIONAL - BOBINA																									
3																										
4																										
5																										
6	Larg. das Bobinas:	50 cm																								
7	Estoque Máximo:	10																								
8																										
9																										
10																										
11																										
12																										
13																										
14																										
15																										
16																										
17																										
18	SOLUÇÃO																									
19																										
20																										
21																										
22																										
23																										
24																										
25																										
26																										
27																										
28																										
29																										
30																										
31																										
32																										
33																										
34																										
35																										

Title: CorteBobina(R).lg4;

sets:

```

padroes/@ole('CorteBobina(R).xls','padroes')/: desp, x;
bobinas/@ole('CorteBobina(R).xls','bobinas')/: demanda;
matriz(padroes,bobinas): a;
endsets

```

data:

```

! Importa dados do Excel;
desp, demanda, a, estmax =
@ole('CorteBobina(R).xls', 'desp', 'demanda', 'a', 'estmax');
enddata

! Minimiza os desperdícios produzidos com a utilização dos padrões de corte;
[fo] min = @sum(padroes(i): desp(i)*x(i));

! O total de bobinas de cada tipo produzido deve ser  $\geq$  ao demandado;
@for(bobinas(j): [excesso]@sum(padroes(i): a(i,j)*x(i)) >= demanda(j));

! O total de bobinas de cada tipo produzidas deve ser  $\leq$  ao demandado
mais o permitido em estoque;
@for(bobinas(j): @sum(padroes(i): a(i,j)*x(i)) <= demanda(j) + estmax);

! A variável x(i) deve ser inteira;
@for(padroes(i): @GIN(x(i)));

```

data:

```

! Exportando dados para o Excel;
@ole('CorteBobina(R).xls', 'excesso', 'solucao', 'ptotal') =
excesso, x, fo;
enddata

```

- (12) **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.

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 ;
 $demandaj$: Demanda por barras do tipo j ;
 a_{ij} : Quantidade de barras do tipo j produzidas no padrão i ;

e a seguinte variável de decisão:

x_i : Número de cortes realizados segundo o padrão i

O modelo de programação matemática para o Exercício 12 é:

$$\min \sum_{i \in padroes} (perda_i \times x_i)$$

s.a:

$$\sum_{i \in padroes} a_{ij}x_i \geq demanda_j \quad \forall j \in barras$$

$$x_i \in \mathbb{Z}^+ \quad \forall i \in padroes$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														

CORTE UNIDIMENSIONAL

Tam. das Barras: **7 m**

SOLUÇÃO

Padrão **Barra (m)** **barras**

	2	3	4	Perda (m)
Padrão	B2	B3	B4	
1	0	0	1	3
2	0	1	0	4
3	0	1	1	0
4	0	2	0	1
5	1	0	0	5
6	1	1	0	2
7	1	0	1	1
8	2	0	0	3
9	2	1	0	0
10	3	0	0	1

padroes a perda

Demandas **Demanda** **excesso**

	92	59	89
Demandas			

demandas excesso

Padrão **Barra (m)** **N. Cortes**

	2	3	4	N. Cortes
Padrão	B2	B3	B4	
1	0	0	1	0
2	0	1	0	0
3	0	1	1	89
4	0	2	0	0
5	1	0	0	0
6	1	1	0	0
7	1	0	1	0
8	2	0	0	0
9	2	1	0	46
10	3	0	0	0
Atendido	92	135	89	
Excesso	0	76	0	solucao

excesso solucao

Perda Total: **0** **ptotal**

ptotal

Title: Corte(R).lg4;

sets:

padroes/@ole('Corte(R).xls','padroes')/: perda, x;

barras/@ole('Corte(R).xls','barras')/: demanda;

matriz(padroes,barras): a;

endsets

data:

! Importa dados do Excel;

perda, demanda, a = @ole('Corte(R).xls','perda','demanda','a');

enddata

! Minimiza a perda com os cortes;

[fo] min = @sum(padroes(i): perda(i)*x(i));

! Total de barras produzidas dever ser \geq ao demandado;

@for(barras(j): [excesso] @sum(padroes(i): a(i,j)*x(i)) \geq demanda(j));

! A variável $x(i)$ deve ser inteira;

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

data:

! Exportando dados para o Excel;

@ole('Corte(R).xls','excesso','solucao','ptotal') = excesso, x, fo;

enddata

- (13) **Problema de Corte de Estoque Unidimensional** 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.

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$padroes$: Conjunto de padrões de corte;
 $barras$: Conjunto de barras;
 $perda_i$: Perda com os cortes realizados de acordo com o padrão i , em metros;
 $demandaj$: Demanda por barras do tipo j ;
 $dimenbarra_j$: Dimensão da barra do tipo j , em metros;
 a_{ij} : Quantidade de barras do tipo j produzidas no padrão de corte i ;

e a seguinte variável de decisão:

x_i : Número de cortes realizados segundo o padrão i

O modelo de programação matemática para o Exercício 13 é:

$$\min \sum_{i \in padroes} (perda_i \times x_i) + \sum_{j \in barras} (dimenbarra_j \times (\sum_{i \in padroes} a_{ij}x_i - demanda_j))$$

s.a:

$$\sum_{i \in padroes} a_{ij}x_i \geq demanda_j \quad \forall j \in barras$$

$$x_i \in \mathbb{Z}^+ \quad \forall i \in padroes$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														

Title: CorteExcesso(R).lg4;

sets:

```

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

```

data:

```

! Importando dados do Excel;
perda, demanda, a, dimenbarra =
@ole('CorteExcesso(R).xls','perda','demanda','a','dimenbarra');
enddata

```

! Minimiza as perdas com os cortes e o excesso de barras não aproveitadas;

```

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

```

! Total de barras produzidas dever ser \geq ao demandado;

```

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

```

! A variável x(i) deve ser inteira;

```

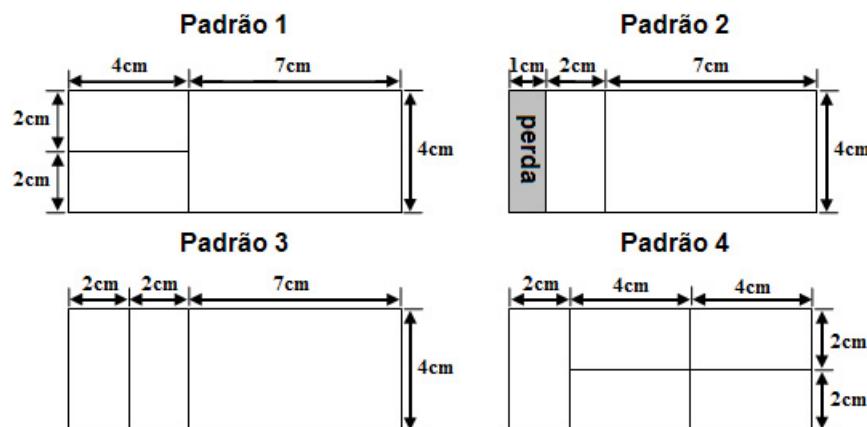
@for(padroes(i): @GIN(x(i)));

```

data:

```
! Exportando dados para o Excel;
@ole('CorteExcesso(R).xls','excesso','solucao','ptotal') =
excesso, x, fo;
enddata
```

- (14) **Problema de Corte de Estoque Bidimensional** Um fabricante de tiras metálicas recebeu um pedido para produzir 2000 tiras de tamanho $2 \text{ cm} \times 4 \text{ cm}$ e 1000 tiras de $4 \text{ cm} \times 7 \text{ cm}$. As tiras podem ser produzidas a partir de chapas maiores disponíveis nos tamanhos de $10 \text{ cm} \times 3000 \text{ cm}$ e $11 \text{ cm} \times 2000 \text{ cm}$. O departamento técnico encarregado de planejar o atendimento ao pedido decidiu que os padrões de corte mostrados na figura a seguir são adequados para produzir as tiras encomendadas. Formule um modelo de programação linear que permita minimizar a quantidade de material usado (ou, equivalentemente, minimizar as perdas) para o atendimento da encomenda.



- (15) **Problema da Mochila 0-1** Há um conjunto de 10 itens e uma mochila com capacidade igual a 100 u.m. Sabendo-se que existe uma única unidade de cada item e, que a cada item está associado um peso e um valor, conforme tabela abaixo, formule um modelo que maximize o valor total dos itens alocados à mochila.

Item	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

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $itens$: Conjunto de itens;
 $peso_j$: Peso associado ao item j ;
 $valor_j$: Valor associado ao item j ;
 cap : Capacidade da mochila;

e a seguinte variável de decisão:

$$x_j : \begin{cases} 1 & \text{se o item } j \text{ for alocado à mochila;} \\ 0 & \text{caso contrário.} \end{cases}$$

O modelo de programação matemática para o Exercício 15 é:

$$\max \sum_{j \in itens} (valor_j \times x_j)$$

s.a:

$$\begin{aligned} \sum_{j \in itens} peso_j \times x_j &\leq cap \\ x_j &\in \{0, 1\} \quad \forall j \in itens \end{aligned}$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q																																	
1	MOCHILA 0-1																																																	
2																																																		
3																																																		
4																																																		
5																																																		
6	Capac. da Mochila: 100			<table border="1"> <thead> <tr> <th>Item</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th> </tr> </thead> <tbody> <tr> <td>Peso (kg)</td><td>15</td><td>18</td><td>13</td><td>23</td><td>9</td><td>10</td><td>11</td><td>5</td><td>14</td><td>5</td> </tr> <tr> <td>Valor</td><td>5</td><td>7</td><td>6</td><td>10</td><td>8</td><td>3</td><td>4</td><td>1</td><td>7</td><td>3</td> </tr> </tbody> </table>														Item	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
Item	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																																								
7	↑ cap			← itens ← peso ← valor																																														
8																																																		
9																																																		
10																																																		
11																																																		
12																																																		
13																																																		
14																																																		
15																																																		
16	SOLUÇÃO																																																	
17																																																		
18																																																		
19				<table border="1"> <thead> <tr> <th>Item</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th> </tr> </thead> <tbody> <tr> <td>Levado</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td> </tr> </tbody> </table>														Item	1	2	3	4	5	6	7	8	9	10	Levado	1	1	1	1	1	0	0	0	1	1											
Item	1	2	3	4	5	6	7	8	9	10																																								
Levado	1	1	1	1	1	0	0	0	1	1																																								
20				← solucao																																														
21																																																		
22																																																		
23																																																		
24																																																		
25																																																		
26																																																		
27																																																		
28																																																		
29																																																		
30																																																		
31																																																		
32																																																		
33																																																		
34																																																		
35																																																		

Valor Máximo: 46
 ↑
 valormax

Title: Mochila0-1(R).lg4;

sets:

```
itens/@ole('Mochila0-1(R).xls','itens')/: peso, valor, x;
endsets
```

data:

! Importa dados do Excel;

```
peso, valor, cap = @ole('Mochila0-1(R).xls','peso','valor','cap');
enddata
```

! Maximiza o benefício com os itens alocados à mochila;

```
[fo] max = @sum(itens(j): valor(j)*x(j));
```

! Peso total dos itens alocados à mochila não deve ultrapassar a capacidade da mesma;

```
@sum(itens(j): peso(j)*x(j)) <= cap;
```

! A variável x(j) deve ser binária;

```
@for(itens(j): @BIN(x(j)));
```

data:

! Exporta dados para o Excel;

```
@ole('Mochila0-1(R).xls','solucao','valormax') = x, fo;
```

enddata

- (16) **Problema da Mochila 0-1 Múltipla** Resolva o problema anterior, supondo que existem várias mochilas, cujas capacidades estão representadas nas tabelas a seguir.

Item	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

Mochila	Capacidade
A	47
B	28
C	42

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- itens* : Conjunto de itens;
- mochilas* : Conjunto de Mochilas;
- peso_j* : Peso associado ao item *j*;
- valor_j* : Valor associado ao item *j*;
- cap_i* : Capacidade da mochila *i*;

e a seguinte variável de decisão:

$$x_{ij} : \begin{cases} 1 & \text{se o item } j \text{ for alocado à mochila } i; \\ 0 & \text{caso contrário.} \end{cases}$$

O modelo de programação matemática para o Exercício 16 é:

$$\max \sum_{i \in \text{mochilas}} \sum_{j \in \text{itens}} (\text{valor}_j \times x_{ij})$$

s.a:

$$\sum_{i \in \text{mochilas}} x_{ij} \leq 1 \quad \forall j \in \text{itens}$$

$$\sum_{j \in \text{itens}} \text{peso}_j \times x_{ij} \leq \text{cap}_i \quad \forall i \in \text{mochilas}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \text{mochilas}, \forall j \in \text{itens}$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q																																																					
1	MOCHILA 0-1 MÚLTIPLA																																																																					
2																																																																						
3																																																																						
4																																																																						
5																																																																						
6	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>Item</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> <td>8</td> <td>9</td> <td>10</td> </tr> <tr> <td>Peso (kg)</td> <td>15</td> <td>18</td> <td>13</td> <td>23</td> <td>9</td> <td>10</td> <td>11</td> <td>5</td> <td>14</td> <td>5</td> </tr> <tr> <td>Valor</td> <td>5</td> <td>7</td> <td>6</td> <td>10</td> <td>8</td> <td>3</td> <td>4</td> <td>1</td> <td>7</td> <td>3</td> </tr> </table> ←itens ←peso ←valor																Item	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																					
Item	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																																																												
7																																																																						
8																																																																						
9																																																																						
10																																																																						
11	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>Mochila</td> <td>Capacidade</td> </tr> <tr> <td>A</td> <td>47</td> </tr> <tr> <td>B</td> <td>28</td> </tr> <tr> <td>C</td> <td>42</td> </tr> </table> ↑ mochilas ↑ cap																Mochila	Capacidade	A	47	B	28	C	42																																														
Mochila	Capacidade																																																																					
A	47																																																																					
B	28																																																																					
C	42																																																																					
12																																																																						
13																																																																						
14																																																																						
15																																																																						
16	SOLUÇÃO																																																																					
17																																																																						
18																																																																						
19																																																																						
20	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td rowspan="2">Mochila</td> <td colspan="10">Item</td> </tr> <tr> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> <td>8</td> <td>9</td> <td>10</td> </tr> <tr> <td>A</td> <td>0</td> <td>1</td> <td>1</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>B</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>C</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> </table> ↑ solucao																Mochila	Item										1	2	3	4	5	6	7	8	9	10	A	0	1	1	0	1	0	0	0	0	1	B	1	0	0	0	0	0	1	0	0	0	C	0	0	0	1	0	0	0	1	1	0
Mochila	Item																																																																					
	1	2	3	4	5	6	7	8	9	10																																																												
A	0	1	1	0	1	0	0	0	0	1																																																												
B	1	0	0	0	0	0	1	0	0	0																																																												
C	0	0	0	1	0	0	0	1	1	0																																																												
21																																																																						
22																																																																						
23																																																																						
24																																																																						
25																																																																						
26																																																																						
27																																																																						
28																																																																						
29																																																																						
30																																																																						
31																																																																						
32																																																																						
33																																																																						
34																																																																						
35																																																																						

Valor Máximo: 51
↑
valormax

Title: Mochila0-1Multipla(R).lg4;

sets:

```
mochilas/@ole('Mochila0-1Multipla(R).xls','mochilas')/: cap;
itens/@ole('Mochila0-1Multipla(R).xls','itens')/: peso, valor;
matriz(mochilas,itens): x;
endsets
```

data:

```
! Importa dados do Excel;
peso, valor, cap = @ole('Mochila0-1Multipla(R).xls','peso','valor','cap');
enddata
```

! Maximiza o benefício com a alocação de itens às mochilas

```
[fo] max = @sum(matriz(i,j): valor(j)*x(i,j));
```

! Cada item só pode ser alocado a uma única mochila;

```
@for(itens(j): @sum(mochilas(i): x(i,j)) <= 1);
```

! A capacidade da mochila deve ser respeitada;

```
@for(mochilas(i): @sum(itens(j): peso(j)*x(i,j)) <= cap(i));
```

! A variável $x(i,j)$ deve ser binária;

```
@for(matriz(i,j):@BIN(x(i,j)));
```

data:

```
! Exporta dados para o Excel;
@ole('Mochila0-1Multipla(R).xls','solucao','valormax') = x, fo;
enddata
```

- (17) **Problema da Mochila Inteira** Considere um conjunto de 10 itens e uma mochila de capacidade igual a 100 u.m. Sabendo-se que existem várias unidades de cada item e que a cada um deles está associado um determinado peso e valor, conforme tabela abaixo, formule um modelo que maximize o valor total dos itens alocados à mochila.

Item	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
Unidades	2	3	2	3	2	2	2	2	1	2

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $itens$: Conjunto de itens;
 $peso_j$: Peso associado ao item j ;
 $valor_j$: Valor associado ao item j ;

$quant_j$: Quantidade disponível do item j ;
 cap : Capacidade da mochila;

e a seguinte variável de decisão:

x_i : Quantidade de itens do tipo j a serem alocados à mochila

O modelo de programação matemática para o Exercício 17 é:

$$\max \sum_{j \in \text{itens}} (\text{valor}_j \times x_j)$$

s.a:

$$x_j \leq quant_j \quad \forall j \in \text{itens}$$

$$\sum_{j \in \text{itens}} peso_j \times x_j \leq cap$$

$$x_j \in \mathbb{Z}^+ \quad \forall j \in \text{itens}$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	MOCHILA INTEIRA																	
2																		
3																		
4																		
5	Capac. da Mochila:	100																
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16	SOLUÇÃO																	
17																		
18																		
19																		
20																		
21																		
22																		
23																		
24																		
25																		
26																		
27																		
28																		
29																		
30																		
31																		
32																		
33																		
34																		
35																		

Item	1	2	3	4	5	6	7	8	9	10	Peso	Valor
Levado	0	0	2	2	2	0	0	0	0	2	100	54

↑
solucao

Valor Máximo: 54
↑
valormax

Title: MochilaInteira(R).lg4;

sets:

```
itens/@ole('MochilaInteira(R).xls','itens')/: peso, valor, quant, x;
endsets
```

data:

! Importa dados do Excel;

```
peso, valor, quant, cap = @ole('MochilaInteira(R).xls','peso','valor','quant','cap');
enddata
```

! Maximiza o benefício com a alocação de itens à mochila;

```
[fo] max = @sum(itens(j): valor(j)*x(j));
```

! O número de itens alocados não pode superar a quantidade disponível;

```
@for(itens(j): x(j) <= quant(j));
```

! A capacidade da mochila deve ser respeitada;

```
@sum(itens(j): peso(j)*x(j)) <= cap;
```

! A variável x(j) deve ser inteira;

```
@for(itens(j):@GIN(x(j)));
```

data:

! Exporta dados para o Excel;

```
@ole('MochilaInteira(R).xls','solucao','valormax') = x, fo;
```

```
enddata
```

- (18) **Problema da Mochila Inteira Múltipla** Resolva o problema anterior, supondo que existem vários itens e várias mochilas. Os dados para o problema estão indicados nas tabelas a seguir.

Item	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
Unidades	2	3	2	3	2	2	2	2	1	2

Mochila	Capacidade
A	47
B	28
C	42

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

itens : Conjunto de itens;

- $mochilas$: Conjunto de Mochilas;
 $peso_j$: Peso associado ao item j ;
 $valor_j$: Valor associado ao item j ;
 $quant_j$: Quantidade disponível do item j ;
 cap_i : Capacidade da mochila i ;

e a seguinte variável de decisão:

x_{ij} : Quantidade de itens do tipo j alocados à mochila i

O modelo de programação matemática para o Exercício 18 é:

$$\max \sum_{i \in mochilas} \sum_{j \in itens} (valor_j \times x_{ij})$$

s.a:

$$\sum_{i \in mochilas} x_{ij} \leq quant_j \quad \forall j \in itens$$

$$\sum_{j \in itens} peso_j \times x_{ij} \leq cap_i \quad \forall i \in mochilas$$

$$x_{ij} \in \mathbb{Z}^+ \quad \forall i \in mochilas, \forall j \in itens$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
MOCHILA INTEIRA MÚLTIPLA																	
1																	
2																	
3																	
4																	
5																	
6																	
7																	
8																	
9																	
10																	
11																	
12																	
13																	
14																	
15																	
16																	
17																	
18																	
19																	
20																	
21																	
22																	
23																	
24																	
25																	
26																	
27																	
28																	
29																	
30																	
31																	
32																	
33																	
34																	
35																	

Mochila	Item										Peso	Valor
	1	2	3	4	5	6	7	8	9	10		
A	0	0	0	1	1	0	0	0	1	0	46	25
B	0	0	0	1	0	0	0	0	0	1	28	13
C	0	0	2	0	1	0	0	0	0	1	40	23

solucao

Total	0	0	2	2	2	0	0	0	1	2

Valor Máximo: 61
valormax

Title: MochilaInteiraMultipla(R).lg4;

sets:

```
mochilas/@ole('MochilaInteiraMultipla(R).xls','mochilas')/: cap;
itens/@ole('MochilaInteiraMultipla(R).xls','itens')/: peso, valor, quant;
matriz(mochilas,itens): x;
endsets
```

data:

```
! Importa dados do Excel;
peso, valor, quant, cap = @ole('MochilaInteiraMultipla(R).xls','peso','valor','quant','cap');
enddata
```

! Maximiza o benefício com a alocação dos itens às mochilas;

```
[fo] max = @sum(matriz(i,j): valor(j)*x(i,j));
```

! A quantidade de itens alocados dever ser \leq ao total de itens disponíveis;

```
@for(itens(j): @sum(mochilas(i): x(i,j)) <= quant(j));
```

! A capacidade da mochila deve ser respeitada;

```
@for(mochilas(i): @sum(itens(j): peso(j)*x(i,j)) <= cap(i));
```

! A variável $x(i,j)$ deve ser inteira;

```
@for(matriz(i,j): @GIN(x(i,j)));
```

data:

! Exporta dados para o Excel;

```
@ole('MochilaInteiraMultipla(R).xls','solucao','valormax') = x, fo;
```

enddata

- (19) **Problema de Localização** Há um conjunto de m possíveis localidades para instalar uma fábrica. Há também um conjunto de n clientes a serem atendidos pelas fábricas. Para cada possível fábrica i é dado o custo fixo de produção f_i , sua capacidade de produção cap_i e o custo de transporte c_{ij} de uma unidade do produto da fábrica i para o cliente j . Para cada cliente j é dada a demanda d_j . Em quais localidades deve-se instalar as fábricas de forma a atender a demanda requerida no menor custo?

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

$clientes$:	Conjunto de clientes;
$fabricas$:	Conjunto de fábricas;
f_i	:	Custo fixo de produção na fábrica i ;
cap_i	:	Capacidade de produção da fábrica i ;
d_j	:	Demanda do cliente j ;
c_{ij}	:	Custo de transporte de uma unidade do produto da fábrica i para o cliente j ;

e as seguintes variáveis de decisão:

- x_{ij} : Quantidade de produtos a serem transportados da fábrica i para o cliente j
 y_i : 1 se a fábrica i for instalada e 0, caso contrário

O modelo de programação matemática para o Exercício 19 é:

$$\min \sum_{i \in \text{fabricas}} \sum_{j \in \text{clientes}} c_{ij} x_{ij} + \sum_{i \in \text{fabricas}} f_i y_i$$

s.a:

$$\begin{aligned} \sum_{i \in \text{fabricas}} x_{ij} &\geq d_j & \forall j \in \text{clientes} \\ \sum_{j \in \text{clientes}} x_{ij} &\leq cap_i y_i & \forall i \in \text{fabricas} \\ x_{ij} &\in \mathbb{Z}^+ & \forall i \in \text{fabricas}, \forall j \in \text{clientes} \\ y_i &\in \{0, 1\} & \forall i \in \text{fabricas} \end{aligned}$$

- (20) **Caixeiro Viajante** Considere um conjunto de cidades espalhadas geograficamente. Conhecendo-se as distâncias d_{ij} entre cada par de cidades e sabendo-se que a matriz de distâncias entre elas é simétrica, determine a rota a ser seguida por um vendedor que saia da cidade origem (Atlanta), passe por todas as demais cidades uma única vez e retorne à cidade origem ao final do percurso percorrendo a menor distância possível.

Cidade	Atlanta	Chicago	Cincinnati	Houston	Los Angeles
Atlanta	-	702	454	842	2396
Chicago	702	-	324	1093	2136
Cincinnati	454	324	-	1137	2180
Houston	842	1093	1137	-	1616
Los Angeles	2396	2136	2180	1616	-

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $cidades$: Conjunto de cidades;
 $dist_{ij}$: Distância entre as cidades i e j .

e as seguintes variáveis de decisão:

- x_{ij} : $\begin{cases} 1 & \text{se a aresta } (i,j) \text{ for usada} \\ 0 & \text{caso contrário.} \end{cases}$
 f_{ij} : Fluxo de i para j .

O modelo de programação matemática para o Exercício 20 é:

$$\min \sum_{i \in cidades} \sum_{j \in cidades} dist_{ij} \times x_{ij}$$

s.a:

$$\sum_{i \in cidades} x_{ij} = 1 \quad \forall j \in cidades$$

$$\sum_{j \in cidades} x_{ij} = 1 \quad \forall i \in cidades$$

$$\sum_{i \in cidades} f_{ij} - \sum_{i \in cidades} f_{ji} = 1 \quad \forall j \in cidades \mid j \neq 1$$

$$f_{ij} \leq (|cidades| - 1)x_{ij} \quad \forall i \in cidades, \forall j \in cidades$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in cidades, \forall j \in cidades$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2	CAIXEIRO VIAJANTE													
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														

Cidade	Atlanta	Chicago	Cincinnati	Houston	Los Angeles
Atlanta	0	702	454	842	2396
Chicago	702	0	324	1093	2136
Cincinnati	454	324	0	1137	2180
Houston	842	1093	1137	0	1616
Los Angeles	2396	2136	2180	1616	0

cidades

dist

Cidade	Atlanta	Chicago	Cincinnati	Houston	Los Angeles
Atlanta	0	0	0	1	0
Chicago	0	0	1	0	0
Cincinnati	1	0	0	0	0
Houston	0	0	0	0	1
Los Angeles	0	1	0	0	0

SOLUÇÃO

solucao

Custo Total: 5.372 ← ctotal

Title: CaixeiroViajante(R).lg4;

sets:

```
cidados/@ole('CaixeiroViajante(R).xls','cidados')/;;
matriz(cidados,cidados): dist, x, f;
endsets
```

data:

```
! Importa dados do Excel;
dist = @ole('CaixeiroViajante(R).xls','dist');
enddata
```

! Minimiza a distância total percorrida;

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

! A cada cidade j só chega uma aresta;

```
@for(cidados(j): @sum(cidados(i): x(i,j)) = 1);
```

! De cada cidade i só sai uma aresta;

```
@for(cidados(i): @sum(cidados(j): x(i,j)) = 1);
```

! O fluxo que chega a uma cidade j, exceto a origem, menos o que sai é igual a uma unidade;

```
@for(cidados(j) | j #ne# 1: @sum(cidados(i): f(i,j)) - @sum(cidados(i): f(j,i)) = 1);
```

! O fluxo máximo em cada aresta é n-1, onde n é o número de cidades;

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

! A variável x(i,j) deve ser binária;

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

data:

! Exporta dados para o Excel;

```
@ole('CaixeiroViajante(R).xls','solucao','ctotal') = x, fo;
```

enddata

- (21) **Roteamento de Veículos** Considere um centro de distribuição (depósito) e um conjunto de clientes distribuídos geograficamente. São dadas as distâncias entre o depósito e cada cliente, bem como entre cada par de clientes. A cada cliente está associado uma demanda a ser atendida. Considerando que no depósito há uma frota de veículos de mesma capacidade, quais as rotas a serem seguidas pelos veículos de forma a atender a demanda dos clientes percorrendo a menor distância possível?

Cidade	Depósito	A	B	C	D	E
Depósito	-	996	2162	1067	499	2054
A	996	-	1167	1019	596	1059
B	2162	1167	-	1747	1723	214
C	1067	1019	1747	-	710	1538
D	499	596	1723	710	-	1589
E	2054	1059	214	1538	1589	-
Demandas	0	6	3	7	7	18

Modelo de Programação Matemática

Sejam os seguintes dados de entrada para o problema:

- $cidades$: Conjunto de cidades = { Depósito \cup Clientes };
 $dist_{ij}$: Distância entre as cidades i e j ;
 $demandaj$: Demanda requerida pela cidade j ;
 $capVeic$: Capacidade de carga dos veículos.

e as seguintes variáveis de decisão:

- x_{ij} : $\begin{cases} 1 & \text{se a aresta (i,j) for usada} \\ 0 & \text{caso contrário.} \end{cases}$
 f_{ij} : Fluxo de i para j .

O modelo de programação matemática para o Exercício 21 é:

$$\min \sum_{i \in cidades} \sum_{j \in cidades} dist_{ij} \times x_{ij}$$

s.a:

$$\sum_{j \in cidades} x_{ij} = 1 \quad \forall i \in cidades \mid i \neq 1$$

$$\sum_{i \in cidades} x_{ij} = 1 \quad \forall j \in cidades \mid j \neq 1$$

$$\sum_{i \in cidades} f_{ij} - \sum_{i \in cidades} f_{ji} = demandaj \quad \forall j \in cidades \mid j \neq 1$$

$$\sum_{j \in cidades} x_{1j} = \sum_{j \in cidades} x_{j1}$$

$$f_{ij} \leq capVeic \times x_{ij} \quad \forall i \in cidades, \forall j \in cidades$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in cidades, \forall j \in cidades$$

Modelo Lingo

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2	ROTEAMENTO DE VEÍCULOS														
3															
4															
5	Cap. dos Veículos:	25													
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18	SOLUÇÃO														
19															
20															
21															
22															
23															
24															
25															
26															
27															
28															
29															
30															
31															
32															
33															
34															
35															

Cidade	Depósito	A	B	C	D	E
Depósito	0	996	2162	1067	499	2054
A	996	0	1167	1019	596	1059
B	2162	1167	0	1747	1723	214
C	1067	1019	1747	0	710	1538
D	499	596	1723	710	0	1589
E	2054	1059	214	1538	1589	0

↑ ↑ ↑ ↑ ↑ ← cidades

Demandas	0	6	3	7	7	18
----------	---	---	---	---	---	----

← demanda

SOLUÇÃO

Cidade	Depósito	A	B	C	D	E
Depósito	0	0	1	0	1	0
A	0	0	0	1	0	0
B	0	0	0	0	0	1
C	1	0	0	0	0	0
D	0	1	0	0	0	0
E	1	0	0	0	0	0

↑ ↑ ↑ ↑ ← solucao

N. de Veículos: 2

Distância Total: 7.611

← dtotal

Title: Roteamento de Veículos(R).lg4;

sets:

 cidades/@ole('RoteamentoVeiculos(R).xls','cidades')/: demanda;
 matriz(cidades, cidades): dist, x, f;
 endsets

data:

 ! Importa dados do Excel;

 demanda, dist, capVeic = @ole('RoteamentoVeiculos(R).xls','demanda','dist','capVeic');
 enddata

 ! Minimiza a distância total percorrida;

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

 ! De uma cidade i, exceto o depósito, só sai um único veículo;

 @for(cidades(i) | i #ne# 1: @sum(cidades(i): x(i,j)) = 1);

 ! A uma cidade j, exceto o depósito, só chega um único veículo;

 @for(cidades(j) | j #ne# 1: @sum(cidades(i): x(i,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(j): x(j, 1));

 ! Ao passar por uma cidade j, exceto o depósito, o veículo deve atender a demanda

 dessa cidade, isto é, deve deixar demanda(j) unidades de produto na cidade j;

 @for(cidades(j) | j #ne# 1:

 @sum(cidades(i): f(i,j)) - @sum(cidades(i): f(j,i)) = demanda(j));

 ! A quantidade de fluxo de i para j não pode superar a capacidade do veículo;

 @for(matriz(i,j): f(i,j) <= capVeic*x(i,j));

 ! A variável x(i,j) deve ser binária;

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

data:

 ! Exporta dados para o Excel;

 @ole('RoteamentoVeiculos(R).xls','solucao','dtotal') = x, fo;

enddata