LOGISTICA

Laurea Specialistica-Università di Mantova

Anno accademico 2007-2008

Antonio Tiano

Dipartimento di Informatica e Sistemistica

Università di Pavia

Tel: 0382-985361-Fax:0382-985373

email:antonio@control1.unipv.it

LINGO (LINear Generalized Optimization)

E' un software che consente di :

- Risolvere problemi di ottimizzazione di vario tipo (PL, PQ, PNL) in presenza di vincoli espressi da uguaglianze e/o disuguaglianze

Scrivere i programmi usando una sintassi simbolica orientata alla descrizione di modelli matematici

- Interfacciare altri software di uso comune (EXCEL, Access, etc.)

- Una delle più potenti funzionalità di LINGO è la sua capacità di formulare i problemi di ottimizzazione direttamente in una forma molto simile al linguaggio matematico, consentendo anche a chi non possiede una preparazione matematica particolarmente approfondita di utilizzare e gestire modelli matematici e algoritmi di ottimizzazione notevolmente sofisticati.

- Un altro aspetto interessante è la possibilità di separare i dati dalla formulazione del modello. LINGO infatti in grado di leggere i dati anche da un apposito spreadsheet tipo EXCEL, database o file di testo, in modo tale che il modello risulti indipendente dai dati e risulti pertanto molto più facile apportare modifiche e minimizzare le possibilità di errore quando si imposta il modello matematico.

Sintassi di Lingo

- Per i nomi delle variabili si stabilisce che devono avere al massimo 32 caratteri alfanumerici, iniziando con una lettera seguita da lettere, cifre o _. Il compilatore LINGO non distingue maiuscole e minuscole.

- Tutte le istruzioni devono terminare con un punto e virgola.

- Per dichiarare la funzione obiettivo è sufficiente usare le parole riservate MIN MAX (evidenziate in blu) seguite dal segno =

- I commenti (evidenziati in verde) devono iniziare con il simbolo !

- I file di programmi generati da LINGO hanno l'estensione .LG4

Variabili Libere

- LINGO assume che ogni variabile sia un numero reale positivo o nulle, ovvero si ha un limite inferiore di zero ed un limite superiore di $+\infty$.
- Se si vuole rimuovere il limite inferiore, e consentire che una assegnata variabile possa assumere anche valori reali negativi, si utilizza il comando

@FREE(nome_variabile)

Esempio: @FREE(X);

consente alla variabile X di assumere qualsiasi valore reale.

Variabili ristrette

- E’ possibile restringere I valori assunti da una variabile ai soli interi tramite il comando

@ GIN(nome_variabile);

Esempio : @GIN(X);

impone che i valori della variabile X siano numeri interi

Variabili Limitate

- E' possibile limitare inferiormente e superiormente una assegnata variabile tramite il comando

@ BND(limite_inf, nome_variabile, limite_sup)

Esempio : @BND(-1,X,1);

impone che i valori della variabile X siano compresi tra -1 e 1.

Operatori e Funzioni

- Sono disponibili diversi operatori di tipo aritmetico e logico, ciascuno con un suolivello di priorità, ad es.:

Priorità	**Operatori**
max	#NOT#
	^
	*;/
	+; -
	#EQ#; #NE ;#GT# ;#LT#;#LE#;#AND#;OR#
min	<=;=;>=

Funzioni matematiche

@ABS(X)	Valore assoluto
@COS(X)	Coseno
@SIN(X)	Seno
@TAN(X)	Tangente
@EXP(X)	Esponenziale
@LOG(X)	Logaritmo
@SIGN(X)	Segno
@SMIN(X)	Minimo
@SMAX(X)	Massimo

$$Max\ f(x_1, x_2) = 30x_1 + 50x_2$$

$$3x_1 + 2x_2 \le 18$$

$$x_1 \le 4$$

$$x_2 \le 6$$

$$x_1 \ge 0$$

$$x_2 \ge 0$$

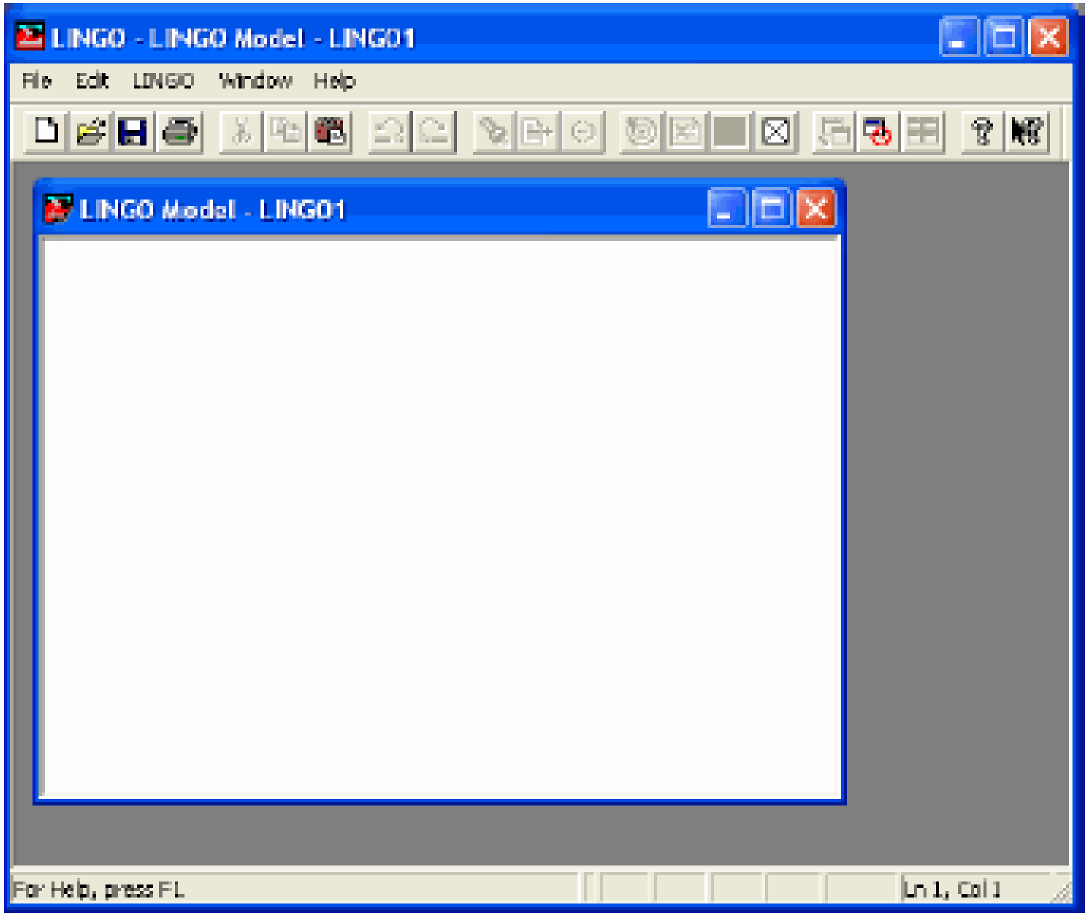
```
MAX=30*X1+50*X2;
3*X1+2*X2<=18;
X1<=4;
X2<=6;
```

Formulazione matematica

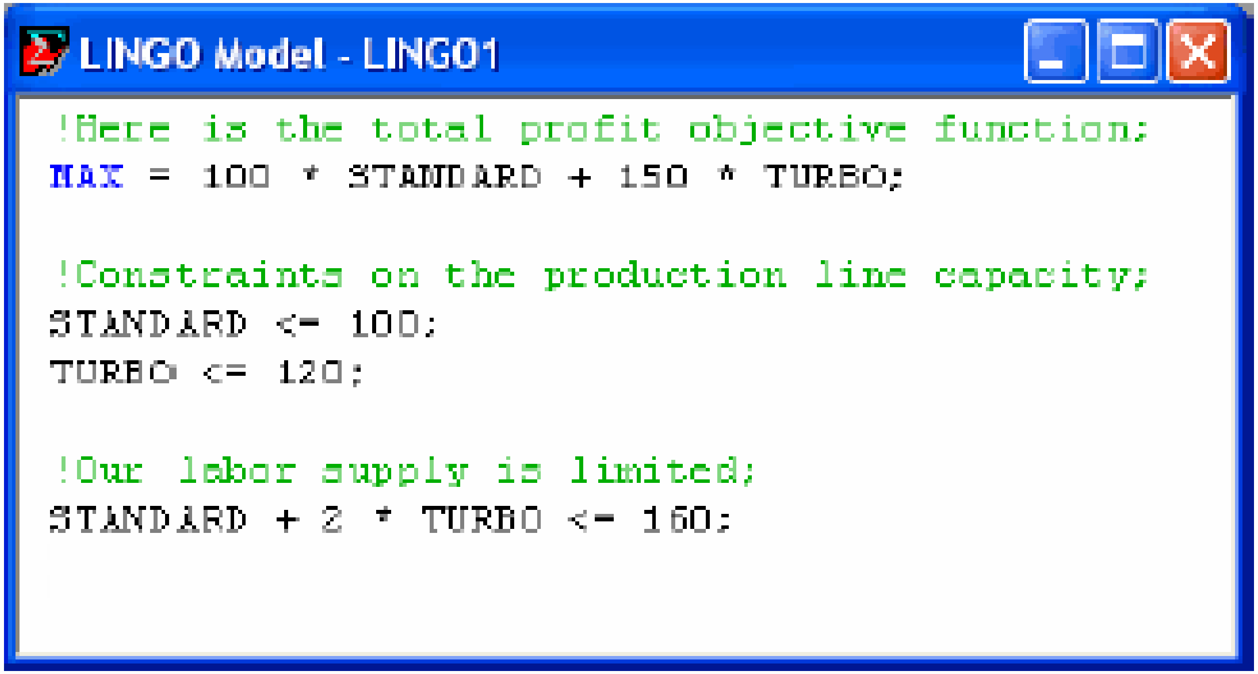
Di un problema di PL

Formulazione

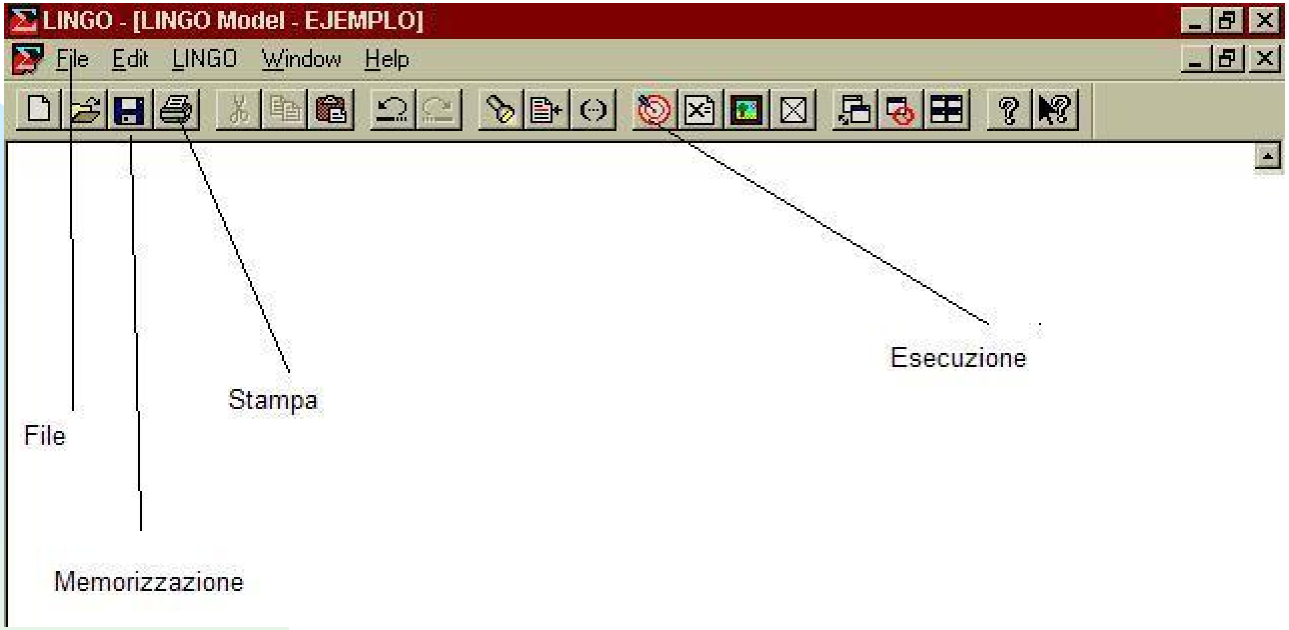
con Lingo



Come introdurre un modello



Esempio di un modello : Programmazione Lineare con 2 variabili decisionali (STANDARD e TURBO) e 3 vincoli lineari



Principali comandi di LINGO eseguibili da menu

```
:   Global optimal solution found.
  Objective value:                              14500.00
  Total solver iterations:                             0

                    Variable           Value        Reduced Cost
                    STANDARD        100.0000            0.000000
                       TURBO        30.00000            0.000000

                         Row    Slack or Surplus      Dual Price
                           1        14500.00            1.000000
                           2        0.000000            25.00000
                           3        90.00000            0.000000
                           4        0.000000            75.00000
```

Variaibili decisionali soluzione del problema di Programmazione Lineare

Ottimo Funzione obiettivo

Finestra della soluzione

Field	Description
Model Class	Displays the model's classification. Possible classes are "LP", "QP", "ILP", "IQP", "PILP", "PIQP", "NLP", "INLP", and "PINLP".
State	Gives the Status of the current solution. Possible states are "Global Optimum", "Local Optimum", "Feasible", "Infeasible", "Unbounded", "Interrupted", and "Undetermined".
Objective	Current value of the objective function.
Infeasibility	Amount constraints are violated by.
Iterations	Number of solver iterations.

Model class field : riassume le proprietà del modello

Abbreviation	Class	Description
LP	Linear Program	All expressions are linear and the model contains no integer restrictions on the variables.
QP	Quadratic Program	All expressions are linear or quadratic, the model is convex, and there are no integer restrictions.
ILP	Integer Linear Program	All expressions are linear, and a subset of the variables is restricted to integer values.
IQP	Integer Quadratic Program	All expressions are either linear or quadratic, the model is convex, and a subset of the variables has integer restrictions.
PILP	Pure Integer Linear Program	All expressions are linear, and all variables are restricted to integer values.
PIQP	Pure Integer Quadratic Program	All expressions are linear or quadratic, the model is convex, and all variables are restricted to integer values.
NLP	Nonlinear Program	At least one of the relationships in the model is nonlinear with respect to the variables.
INLP	Integer Nonlinear Program	At least one of the expressions in the model is nonlinear, and a subset of the variables has integer restrictions. ***In general, this class of model will be very difficult to solve for all but the smallest cases.***
PINLP	Pure Integer Nonlinear Program	At least one of the expressions in the model is nonlinear, and all variables have integer restrictions. ***In general, this class of model will be very difficult to solve for all but the smallest cases.***

Classificazione dei problemi di ottimizzazione solubili con LINGO

Struttura di un modello LINGO piu' complesso

- Nome del modello
- Definizione di indici,variabili, costanti
- Assegnazione valori alle costanti
- Comandi

```
MODEL:
! A 6 Warehouse 8 Vendor Transportation Problem;
SETS:
   WAREHOUSES: CAPACITY;
   VENDORS: DEMAND;
   LINKS( WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
! Here is the data;
DATA:
   !set members;
   WAREHOUSES = WH1 WH2 WH3 WH4 WH5 WH6;
   VENDORS = V1 V2 V3 V4 V5 V6 V7 V8;

   !attribute values;
   CAPACITY = 60 55 51 43 41 52;
   DEMAND = 35 37 22 32 41 32 43 38;
   COST = 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
! The objective;
   MIN = @SUM( LINKS( I, J):
    COST( I, J) * VOLUME( I, J));
! The demand constraints;
   @FOR( VENDORS( J):
    @SUM( WAREHOUSES( I): VOLUME( I, J)) =
     DEMAND( J));
! The capacity constraints;
   @FOR( WAREHOUSES( I):
    @SUM( VENDORS( J): VOLUME( I, J)) <=
     CAPACITY( I));
END
```

- I **SETS** sono classi di elementi dotati di attributi quantificabili.
- Essi possono essere definiti come array o matrici

- (Vedi “LINGO USER MANUAL” e Esempi vari presentati in questo Documento)

Interfacciamento di Lingo con files esterni

1. Copia/Incolla
2. Importazione di file di testo (@FILE)
3. Esportazione di file di testo (@TEXT)
4. Importazione dati da Excel (@OLE)
5. Esportazione dati verso Excel (@OLE)
6. Esportazione di “Summary Report” verso Excel (@OLE)

Interfacciamento di Lingo con files esterni

1. Copia/Incolla

E' possibile eseguire questa operazione da e verso qualsiasi programma che supporta tale comando.

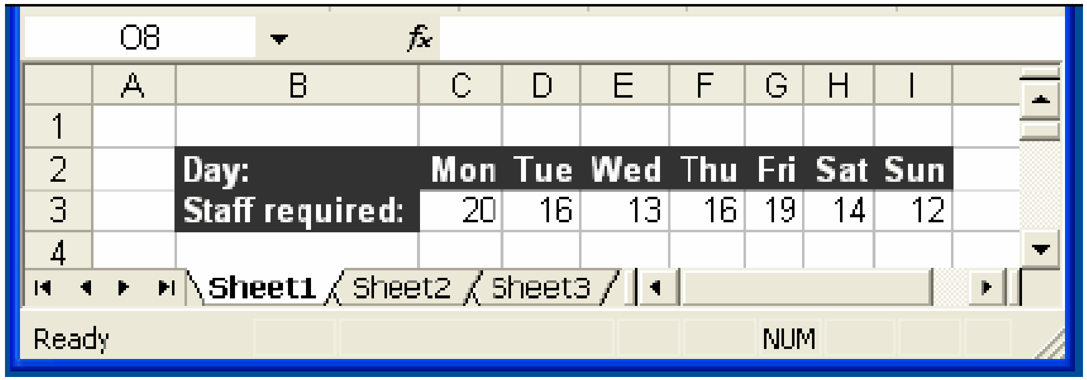
Vedi esempio Pag. 59 "User Manual", al quale si applica

l'importazione di dati da un file esterno Excel

```
SETS:
   DAYS / MON TUE WED THU FRI SAT SUN/:
    REQUIRED, START;
ENDSETS
DATA:
   REQUIRED = <data omitted>;
ENDDATA
MIN = @SUM(DAYS(I): START(I));
@FOR(DAYS(J):
   @SUM(DAYS(I) | I #LE# 5:
      START(@WRAP(J - I + 1, 7)))
         >= REQUIRED(J)
);
```

L'attributo REQUIRED è omesso

Supponendo che i dati mancanti siano contenuti in un file Excel del tipo :



Day:	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Staff required:	20	16	13	16	19	14	12

Per incollare i dati richiesti :

1. Selezionare il range dei dati (C3:I3)
2. Selezionare Copy dal menu di Excel
3. Cliccare sulla finestra del modello LINGO
4. Posizionare il cursore di LINGO a sinistra dell’=
5. Selezionare Paste dall’Editor di LINGO

Il risultato è :

```
DATA:
   REQUIRED = 20 16 13 16 19 14 12;
ENDDATA
```

L’operazione e’ identica per qualunque file tipo .doc, .txt, etc.

Se si vogliono esportare i dati della soluzione da LINGO verso un file di MS Word, la procedura è :

1. Selezionare i dati desiderati in LINGO
2. Selezionare Copy dal menu di LINGO
3. Attivare il documento MS-Word nella posizione in cui si vogliono incollare i dati
4. Selezionare Paste dall'Editor di MS-Word

2. Importazione di file di testo (@FILE)

Si utilizza il comando @FILE, con la sintassi :

@FILE('Nome del file')

Esempio : Modello di trasporto, Pag. 26 "User Manual"

```
MODEL:
! A 6 Warehouse 8 Vendor Transportation Problem;
SETS:
   WAREHOUSES: CAPACITY;
   VENDORS: DEMAND;
   LINKS( WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
! Here is the data;
DATA:
   !set members;
   WAREHOUSES = WH1 WH2 WH3 WH4 WH5 WH6;
   VENDORS = V1 V2 V3 V4 V5 V6 V7 V8;

   !attribute values;
   CAPACITY = 60 55 51 43 41 52;
   DEMAND = 35 37 22 32 41 32 43 38;
   COST = 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
! The objective;
   MIN = @SUM( LINKS( I, J):
    COST( I, J) * VOLUME( I, J));
! The demand constraints;
   @FOR( VENDORS( J):
    @SUM( WAREHOUSES( I): VOLUME( I, J)) =
     DEMAND( J));
! The capacity constraints;
   @FOR( WAREHOUSES( I):
    @SUM( VENDORS( J): VOLUME( I, J)) <=
     CAPACITY( I));
END
```

Dati

Dati

Per separare completamente i dati dal modello, è opportuno importarli da un file esterno tramite il comando @FILE :

```
! A 6 Warehouse 8 Vendor Transportation Problem;
SETS:
   WAREHOUSES / @FILE('WIDGETS2.LDT')/: CAPACITY;
   VENDORS / @FILE('WIDGETS2.LDT')/    : DEMAND;
   LINKS(WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS

! The objective;
   MIN = @SUM(LINKS(I, J):
    COST(I, J) * VOLUME(I, J));

! The demand constraints;
   @FOR(VENDORS(J):
    @SUM(WAREHOUSES(I): VOLUME(I, J)) =
     DEMAND(J));

! The capacity constraints;
   @FOR(WAREHOUSES(I):
    @SUM(VENDORS(J): VOLUME(I, J)) <=
     CAPACITY(I));

! Here is the data;
DATA:
   CAPACITY = @FILE('WIDGETS2.LDT');
   DEMAND = @FILE('WIDGETS2.LDT');
   COST = @FILE('WIDGETS2.LDT');
ENDDATA
```

Il file da importare ha convenzionalmente l'estensione .ldt

Contiene gli attributi di 5 sets di dati

Il contenuto de file WIDGETS2.LDT è il seguente :

```
!List of warehouses;
WH1 WH2 WH3 WH4 WH5 WH6 ~

!List of vendors;
V1 V2 V3 V4 V5 V6 V7 V8 ~

!Warehouse capacities;
60 55 51 43 41 52 ~

!Vendor requirements;
35 37 22 32 41 32 43 38 ~

!Unit shipping costs;
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
```

- I commenti (!) sono ignorati nell’importazione dei dati
- Un modello può importare un max di 16 files esterni

3. Esportazione di file di testo (@TEXT)

Si utilizza il comando @TEXT per esportare le soluzioni verso files di testo, in base alla sintassi :

@TEXT([‘filename’, [‘a’]])

Se gli argomenti ‘filename’ e ‘a’ sono entrambi assenti, il file viene inviato su video.

Se l’argomento ‘a’ è assente, il file ‘filename’ viene creato come nuovo, eliminando qualunque file avesse tale nome.

Se l’argomento [‘a’] è presente, i dati in uscita sono aggiunti al file ‘filename’ .

Esempio 1.

```
@TEXT('RESULTS.TXT') = X;
```

Invia I valori di X al file RESULTS.TXT

Esempio 2.

```
@TEXT() = DAYS, START;
```

Sono inviati su video i valori dei sets DAYS e degli attributi START

Esempio 3.

Example 3:

```
@TEXT() = @WRITEFOR( DAYS( D) | START( D) #GT# 0:
   DAYS( D), '  ', START( D));
```

Sono inviati su video solo i sets DAYS il cui attributo START è >0

Esempio 4.

```
SETS:
   DAYS / MON TUE WED THU FRI SAT SUN/:
    REQUIRED, START;
ENDSETS

DATA:
   REQUIRED = 20 16 13 16 19 14 12;
   @TEXT('OUT.TXT') = DAYS, START;
ENDDATA

MIN = @SUM(DAYS(I): START(I));

@FOR(DAYS(J):
   @SUM(DAYS(I) | I #LE# 5:
      START(@WRAP(J - I + 1, 7)))
         >= REQUIRED(J)
);
```

Sono inviati in 'out.txt'

i sets DAYS e gli attributi START

Files in LINGO

Extension	Description
.LG4	model files (Windows only)
.LNG	model files in text format
.LTF	script files
.LDT	included data files
.LRP	report files

Con sistema operativo diverso da Windows

4. Importazione dati da Excel

Si utilizza il comando @OLE per importare da Excel in LINGO :

Elementi di SETS : in formato text

Attributi di SETS : in formato numerico

E’ possibile importare sia arrays (righe o colonne) che matrici (righe e colonne).

La sintassi è:

object_list = *@OLE*([‘*spreadsheet_file*’] [, *range_name_list*]);

object_list è la lista dei nomi delle variabili da estrarre dallo spreadsheet_file .

Sono possibili tre situazioni diverse:

1) L'argomento di @OLE è omesso, ad esempio :

```
COST, CAPACITY = @OLE();
```

Poichè non è stato specificato alcun file ne' i ranges relativi alle variabili COST e CAPACITY, viene usato uno qualsiasi dei files Excel al momento aperti e alle variabili sono assegnati i valori trovati.

2) L'argomento di @OLE è specificato con un singolo range, ad esempio :

```
COST, CAPACITY = @OLE( 'SPECS.XLS', 'DATATABLE');
```

Agli attributi COST e CAPACITY sono assegnate le due colonne del range DATATABLE del file SPECS.XLS.

3. L’argomento di @OLE è specificato con più ranges, ad esempio :

```
COST, CAPACITY = @OLE( 'SPECS.XLS', 'COST01', 'CAP01');
```

Agli attributi COST e CAPACITY sono assegnati i due ranges contenuti in COST01 E IN CAP01 del file SPECS.XLS.

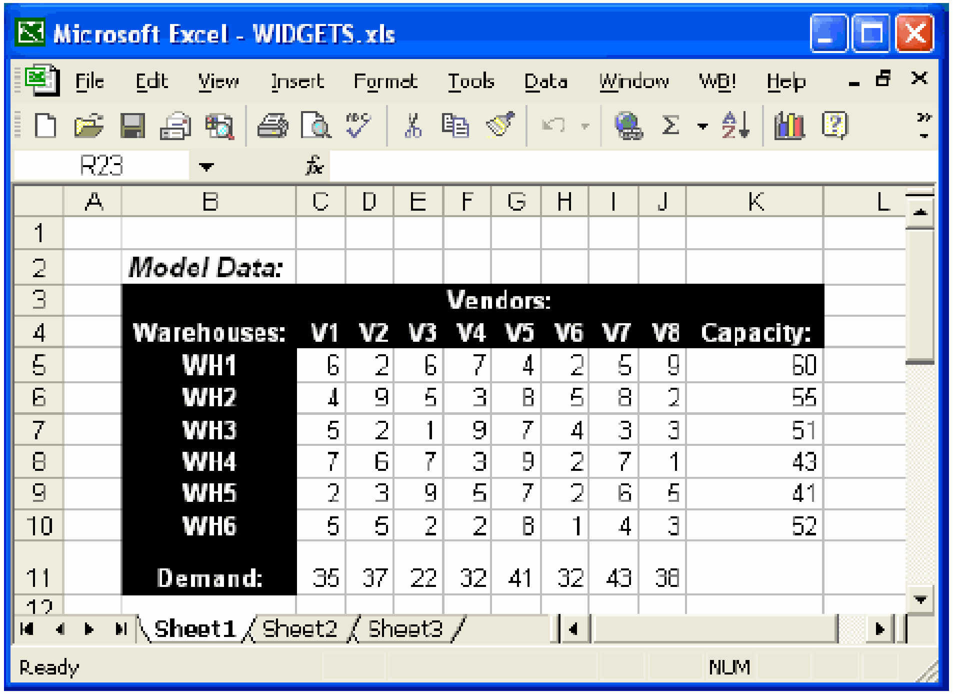
E’ possibile richiamare solo elementi o attributi relativi allo stesso SET, evitando di mescolare elementi e attributi

Esempio 4 : Uso di @OLE in un Modello di trasporto

```
! A 6 Warehouse 8 Vendor Transportation Problem;
SETS:
! Import warehouses and vendors from Excel;
   WAREHOUSES: CAPACITY;
   VENDORS   : DEMAND;
   LINKS(WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS

! The objective;
   MIN = @SUM(LINKS(I, J):
    COST(I, J) * VOLUME(I, J));

! The demand constraints;
   @FOR(VENDORS(J):
    @SUM(WAREHOUSES(I):
     VOLUME(I, J)) = DEMAND(J));

! The capacity constraints;
   @FOR(WAREHOUSES(I):
    @SUM(VENDORS(J): VOLUME(I, J))
     <= CAPACITY(I));

DATA:
! Import the data from Excel;
   WAREHOUSES, VENDORS, CAPACITY, DEMAND, COST =
    @OLE('\LINGO\SAMPLES\WIDGETS.XLS',
     'WAREHOUSES', 'VENDORS', 'CAPACITY',
      'DEMAND', 'COST');
ENDDATA
```

Importazione di dati dal file WIDGETS.XLS

Esempio 4 : Uso di @OLE in un Modello di trasporto



Model Data:

	Vendors:								
Warehouses:	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**Capacity:**
WH1	6	2	6	7	4	2	5	9	60
WH2	4	9	5	3	8	5	8	2	55
WH3	5	2	1	9	7	4	3	3	51
WH4	7	6	7	3	9	2	7	1	43
WH5	2	3	9	5	7	2	6	5	41
WH6	5	5	2	2	8	1	4	3	52
Demand:	35	37	22	32	41	32	43	38	

```
! Import the data from Excel;
  WAREHOUSES, VENDORS, CAPACITY, DEMAND, COST =
   @OLE('\LINGO\SAMPLES\WIDGETS.XLS',
    'WAREHOUSES', 'VENDORS', 'CAPACITY',
     'DEMAND', 'COST');
```

5. Esportazione dati verso Excel

E’ possibile usare due tipi di sintassi:

@OLE(['*spreadsheet_file*'] [, *range_name_list*]) = *object_list*;

object_list è una lista delimitata da virgole di SETS, ATTRIBUTI, e/o variabili scalari (1 dimensione) da esportare.

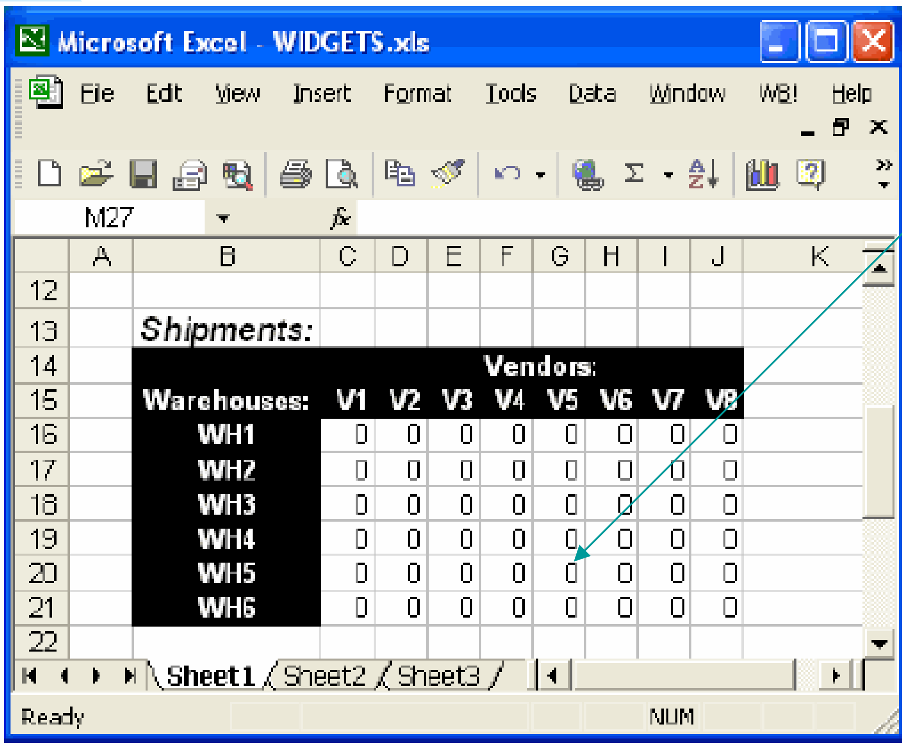
range_name_list è la lista dei ranges nominati di cui si vogliono esportare i valori.

Esempio:Esportazione della soluzione del problema del trasporto

```
! A 6 Warehouse 8 Vendor Transportation Problem;
SETS:
! Import warehouses and vendors from Excel;
   WAREHOUSES: CAPACITY;
   VENDORS    : DEMAND;
   LINKS(WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS

! The objective;
   MIN = @SUM(LINKS(I, J):
    COST(I, J) * VOLUME(I, J));

! The demand constraints;
   @FOR(VENDORS(J):
    @SUM(WAREHOUSES(I):
     VOLUME(I, J)) = DEMAND(J));

! The capacity constraints;
   @FOR(WAREHOUSES(I):
    @SUM(VENDORS(J): VOLUME(I, J))
     <= CAPACITY(I));

DATA:
! Import the data from Excel;
   WAREHOUSES, VENDORS, CAPACITY, DEMAND, COST =
    @OLE('\LINGO\SAMPLES\WIDGETS.XLS',
     'WAREHOUSES', ' VENDORS', 'CAPACITY',
      'DEMAND', 'COST');

! Export the solution back to Excel;
   @OLE('\LINGO\SAMPLES\WIDGETS.XLS',
    'VOLUME') = VOLUME;
ENDDATA
```

Esportazione della soluzione VOLUME

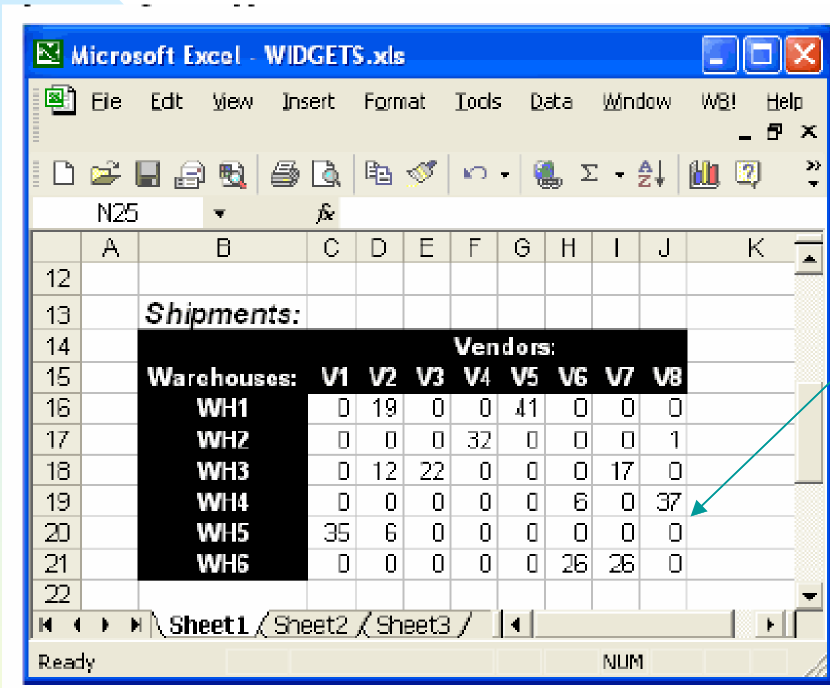
Esempio:Esportazione della soluzione del problema del trasporto



Preparazione dei ranges

Destinati a ricevere i valori della soluzione

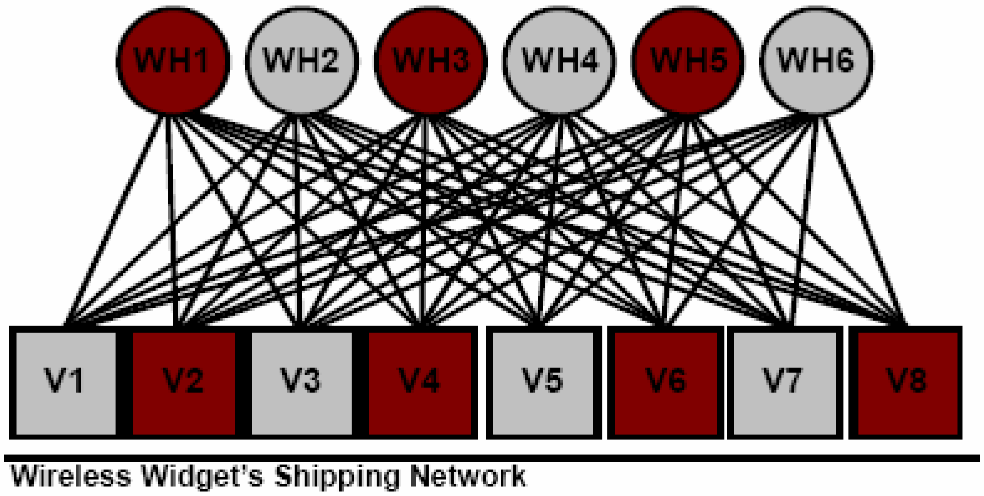
Esempio:Esportazione della soluzione del problema del trasporto



	Vendors:							
Warehouses:	V1	V2	V3	V4	V5	V6	V7	V8
WH1	0	19	0	0	41	0	0	0
WH2	0	0	0	32	0	0	0	1
WH3	0	12	22	0	0	0	17	0
WH4	0	0	0	0	0	6	0	37
WH5	35	6	0	0	0	0	0	0
WH6	0	0	0	0	0	26	26	0

Risultato finale

Problema di trasporto



Si abbiano 6 magazzini WH1,..WH6 che devono rifornire 8 venditori V1,…V8.

Ogni magazzino dispone di una data quantità di merce che non puo' essere superata, mentre ogni venditore richiede una quantità di merce che deve essere consegnata.

Warehouse	Widgets On Hand
1	60
2	55
3	51
4	43
5	41
6	52

Widget Capacity Data

Vendor	Widget Demand
1	35
2	37
3	22
4	32
5	41
6	32
7	43
8	38

Vendor Widget Demand

Vincoli sulle capacità dei magazzini

Vincoli sulle domande dei venditori

	V1	V2	V3	V4	V5	V6	V7	V8
Wh1	6	2	6	7	4	2	5	9
Wh2	4	9	5	3	8	5	8	2
Wh3	5	2	1	9	7	4	3	3
Wh4	7	6	7	3	9	2	7	1
Wh5	2	3	9	5	7	2	6	5
Wh6	5	5	2	2	8	1	4	3

Shipping Cost per Widget ($)

Costo del trasporto tra i magazzini e i venditori

Si vuole minimizzare il costo totale dl trasporto, rispettando i vincoli di capacità e di domanda.

$$x_{ij} = merce\ trasportata\ dal\ magazzino\ i\ al\ venditore\ j$$

$$c_{ij} = costo\ unitario\ del\ trasporto\ da\ i\ a\ j$$

$$a_i = capacità\ del\ magazzino\ i\ \ (i = 1{,}6)$$

$$b_j = domanda\ del\ venditore\ j\ (j = 1{,}8)$$

$$f(x_{11},\dots,x_{18},x_{21},\dots,x_{28},\dots.,x_{61},\dots,x_{68}) = \sum_{i=1}^{6}\sum_{j=1}^{8} c_{ij}x_{ij}$$

Costo totale

$$\sum_{j=1}^{8} x_{ij} \le a_i \quad ; i = 1{,}6$$

Vincoli di capacità

$$\sum_{i=1}^{6} x_{ij} = b_j \quad ; j = 1{,}8$$

Vincoli sulla domanda

Il problema del trasporto si può quindi formulare come un problema di Programmazione Lineare di ricerca del minimo di una funzione obiettivo lineare con :

6x8=48 variabili decisionali

6+8=14 vincoli lineari (equazioni)

Si può risolvere il problema con LINGO

Avendo posto Volume_1_1=x11,.....,Volume_6_8=x68

Dovrei scrivere 48 righe del tipo :

```
MIN = 6 * VOLUME_1_1 + 2 * VOLUME_1_2 +
      6 * VOLUME_1_3 + 7 * VOLUME_1_4 +
      4 * VOLUME_1_5 +
            .
            .
            .
      8 * VOLUME_6_5 + VOLUME_6_6 + 4 * VOLUME_6_7 +
      3 * VOLUME_6_8;
```

E' una operazione noiosa e suscettibile alla introduzione di errori manuali.

Si può procedere in modo più efficiente, tramite l' uso di una notazione vettoriale

La funzione obiettivo si può esprimere come:

MIN = @SUM(LINKS(I,J): COST(I,J) * VOLUME(I,J));

Notazione matematica	Sintassi LINGO
Min	MIN=
$\sum_{i=1}^{6}\sum_{j=1}^{8}$	@SUM(LINKS(I,J):
c_{ij}	COST(I,J)
x_{ij}	VOLUME(I,J)

I vincoli sulla domanda si possono esprimere :

@FOR(VENDORS(J):
@SUM(WAREHOUSES(I): VOLUME(I, J)) =
DEMAND(J));

Notazione matematica	Sintassi LINGO
J=1,8 (per tutti i venditori)	@FOR(VENDORS(J):
$\sum_{i=1}^{6}$	@SUM(WAREHOUSES(I):
x_{ij}	VOLUME(I,J)
b_j	DEMAND(J));

I vincoli di capacità si possono esprimere :

@FOR(WAREHOUSES(I):
@SUM(VENDORS(J): VOLUME(I, J))<=
CAPACITY(I));

Notazione matematica	Sintassi LINGO
i=1,6 (per tutti i magazzini)	@FOR(WAREHOUSES(I):
$\sum_{j=1}^{8}$	@SUM(VENDORS(J):
x_{ij}	VOLUME(I,J)
a_i	CAPACITY(I));

Mettendo tutto insieme, si ottiene la seguente definizione del modello (Model section) del problema di ottimizzazione:

```
MODEL:
MIN = @SUM(LINKS(I, J):
COST(I, J) * VOLUME(I, J));
@FOR(VENDORS(J):
@SUM(WAREHOUSES(I): VOLUME(I, J)) =
DEMAND(J));
@FOR(WAREHOUSES(I):
@SUM(VENDORS(J): VOLUME(I, J)) <=
CAPACITY(I));
END
```

Abbiamo bisogno di definire gli insiemi usati dal modello ("Sets section") e i dati ("Data section").

In tal modo si realizza una separazione del modello dalle sue caratteristiche e dai suoi dati.

I tre insiemi definiti nella sets section sono i seguenti :

```
SETS:
WAREHOUSES: CAPACITY;
VENDORS: DEMAND;
LINKS( WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
```

I sets sono definiti in base ai rispettivi attributi.

L’ultimo set, denominato LINKS, rappresenta i collegamenti nella rete distributiva. Ad ogni collegamento è associato un costo (COST) ed una quantità di merce trasportata (VOLUME).

La sintassi usata per definire questo set differisce dalle precedenti.

Specificando :

LINKS(WAREHOUSES, VENDORS)

si comunica a LINGO che il set LINKS set è derivato dai set WAREHOUSES e VENDORS. In questo caso, LINGO genera tutte le possibili coppie ordinate (magazzino, venditore).

Ciascuna di queste 48 coppie ordinate diventa un membro del set LINKS.

Indice	Arco della rete
1	WH1→V1
2	WH1→V2
....	....
48	WH6→V8

Infine si inserisce la “data section”

```
DATA:
!set members;
WAREHOUSES = WH1 WH2 WH3 WH4 WH5 WH6;
VENDORS = V1 V2 V3 V4 V5 V6 V7 V8;
!attribute values;
CAPACITY = 60 55 51 43 41 52;
DEMAND = 35 37 22 32 41 32 43 38;
COST = 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
```

La separazione della « data section » dalle altre sezioni del programma rende possibile utilizzare lo stesso programma con dati diversi

E’ possibile importare i dati da files in formato diverso (testo, excel, etc.)

Programma completo

```
MODEL:
! A 6 Warehouse 8 Vendor Transportation Problem;
SETS:
WAREHOUSES: CAPACITY;
VENDORS: DEMAND;
LINKS( WAREHOUSES, VENDORS): COST, VOLUME;
ENDSETS
! Here is the data;
DATA:
!set members;
WAREHOUSES = WH1 WH2 WH3 WH4 WH5 WH6;
VENDORS = V1 V2 V3 V4 V5 V6 V7 V8;
!attribute values;
CAPACITY = 60 55 51 43 41 52;
DEMAND = 35 37 22 32 41 32 43 38;
COST = 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
! The objective;
MIN = @SUM( LINKS( I, J):
COST( I, J) * VOLUME( I, J));
! The demand constraints;
@FOR( VENDORS( J):
@SUM( WAREHOUSES( I): VOLUME( I, J)) =
DEMAND( J));
! The capacity constraints;
@FOR( WAREHOUSES( I):
@SUM( VENDORS( J): VOLUME( I, J)) <=
CAPACITY( I));
END
```

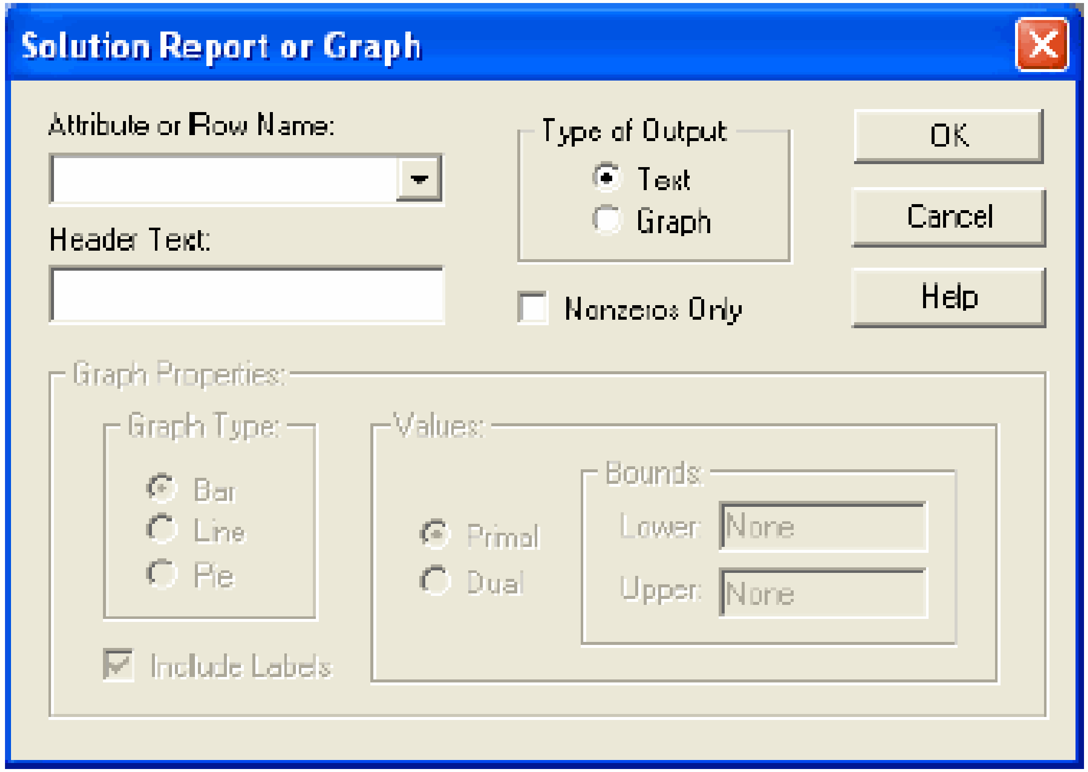
Osservazione

Quando, come in questo caso, l'uscita del programma è "voluminosa", conviene selezionare alcune opzioni di LINGO.

1) Dal menu di LINGO selezionare la **Interface tab**, e settare **Output Level** come **Terse.**
LINGO mostrerà soltanto le soluzioni ed il numero di

2) Per ottenere un report contenente soltanto I valori non nulli dei VOLUME trasportati, si seleziona il comando we select the **Solution**.

Si presenta, allora, la seguente dialog box:

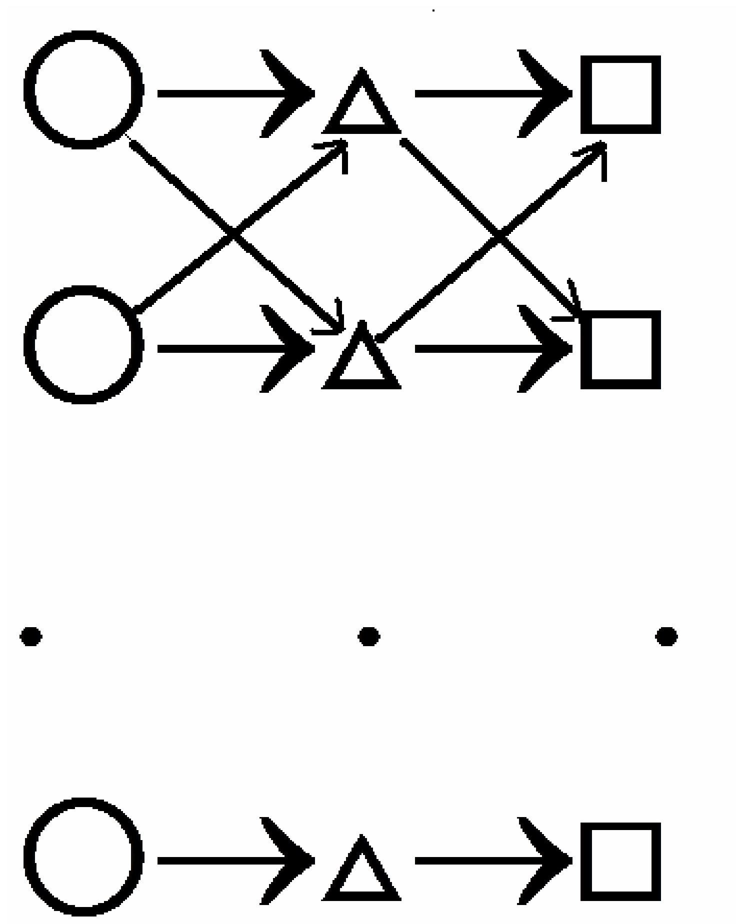


Press down on the arrow button in the Attribute or Row Name field and select VOLUME from the list of names in the drop-down box. To suppress the printing of variables with zero value, click on the Nonzeros Only checkbox. Once you have done this, the dialog box should resemble:

Solution Report - WIDGETS

Variable	Value	Reduced Cost
VOLUME(WH1, V2)	19.00000	0.0000000
VOLUME(WH1, V5)	41.00000	0.0000000
VOLUME(WH2, V1)	1.000000	0.0000000
VOLUME(WH2, V4)	32.00000	0.0000000
VOLUME(WH3, V2)	11.00000	0.0000000
VOLUME(WH3, V7)	40.00000	0.0000000
VOLUME(WH4, V6)	5.000000	0.0000000
VOLUME(WH4, V8)	38.00000	0.0000000
VOLUME(WH5, V1)	34.00000	0.0000000
VOLUME(WH5, V2)	7.000000	0.0000000
VOLUME(WH6, V3)	22.00000	0.0000000
VOLUME(WH6, V6)	27.00000	0.0000000
VOLUME(WH6, V7)	3.000000	0.0000000

Problema del trasporto a due stadi (Transhipment)



Origini Distributori Destinazioni

(i=1,m) (k=1,p) (j=1,n)

Problema del trasporto a due stadi (Transhipment)

x_{ik} = merce trasportata dalla sorgente i al distributore k

y_{kj} = merce trasportata dal distributore k alla destinazione j

c_{ik} = costo unitario del trasporto dalla sorgente i al distributore k

d_{kj} = costo unitario del trasporto dal distributore k alla destinazione j

a_i = capacità della sorgente i

e_k = capacità del distributore k

b_j = domanda della destinazione j

$$i = 1,m;\ k = 1,p;\ j = 1,n$$

$$f(x_{11},\dots,x_{1p},\dots,x_{m1},\dots,x_{mp},y_{11},\dots,y_{1n},\dots,y_{p1},\dots,x_{pn}) = \sum_{i=1}^{m}\sum_{k=1}^{p} c_{ik}x_{ik} + \sum_{k=1}^{p}\sum_{j=1}^{n} d_{kj}y_{kj}$$

$$\sum_{k=1}^{p} x_{ik} \le a_i \quad ;\ i = 1,m \qquad \text{(vincolo di capacità)}$$

$$\sum_{k=1}^{p} y_{kj} = b_j \quad ;\ j = 1,n \qquad \text{(vincolo sulla domanda)}$$

$$\sum_{i=1}^{m} x_{ik} = \sum_{j=1}^{n} y_{kj} \quad ;\ k = 1,p \qquad \text{(vincolo di conservazione del flusso)}$$

Problema del trasporto a due stadi (Transhipment)

Dimensioni del problema:

- **Numero variabili decisionali = $mp+pn=(m+n)p$**
- **Numero vincoli = $m+n+2p$**

Osservazione: Le dimensioni del problema del Transhipment sono diverse dalle dimensioni del problema del trasporto che sono :

- **Numero variabili decisionali = mn**
- **Numero vincoli = $m+n$**

Programma

```
MODEL:
! Un problema di transhipment con 2 Sorgenti 2 Distributori 2 Destinazioni;
SETS:
   SORGENTI: CAPACITA_S;
   DISTRIB;
   DESTINAZIONI: DOMANDE;
   LINKS1(SORGENTI , DISTRIB): COSTI_SD, MERCI_SD;
   LINKS2(DISTRIB, DESTINAZIONI): COSTI_DD,MERCI_DD;
ENDSETS
! Questi sono i  dati;
DATA:
!set members;
   SORGENTI  = S1 S2;
   DISTRIB = D1 D2;
   DESTINAZIONI = DD1 DD2;
   !Valori degli attributi;
   CAPACITA_S = 60 40;
   !CAPACITA_D = 50 50;
   DOMANDE = 35 45 ;
   COSTI_SD = 6 2
              4 9;
   COSTI_DD = 3 5
              6 4;
          ENDDATA
! Funzione obiettivo;
   MIN = @SUM( LINKS1( I, K):
    COSTI_SD( I, K ) * MERCI_SD( I, K))+
         @SUM(LINKS2(K,J):
    COSTI_DD(K,J)*MERCI_DD(K,J));
! Vincoli sulla domanda delle destinazioni;
   @FOR( DESTINAZIONI( J):
    @SUM( DISTRIB( K): MERCI_DD( K, J)) =
     DOMANDE( J));
! Vincoli sulla capacità delle sorgenti;
   @FOR( SORGENTI( I):
    @SUM( DISTRIB( K): MERCI_SD( I, K)) <=
     CAPACITA_S( I));
!  Vincoli di continuità del flusso;
   @FOR(DISTRIB(K):
     @SUM(SORGENTI(I):MERCI_SD(I,K)) =
     @SUM(DESTINAZIONI(J):MERCI_DD(K,J)));
END
```

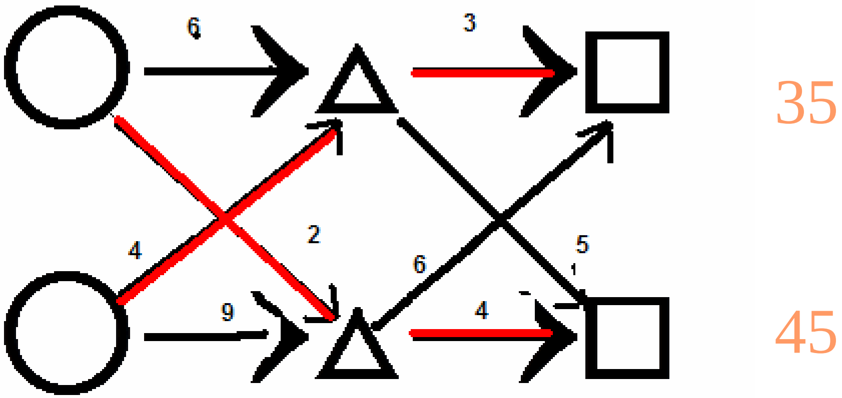
Global optimal solution found.

Objective value: 515.0000

Total solver iterations: 3

Variable	Value	Reduced Cost
CAPACITA_S(S1)	60.00000	0.000000
CAPACITA_S(S2)	40.00000	0.000000
DOMANDE(DD1)	35.00000	0.000000
DOMANDE(DD2)	45.00000	0.000000
COSTI_SD(S1, D1)	6.000000	0.000000
COSTI_SD(S1, D2)	2.000000	0.000000
COSTI_SD(S2, D1)	4.000000	0.000000
COSTI_SD(S2, D2)	9.000000	0.000000
MERCI_SD(S1, D1)	0.000000	2.000000
MERCI_SD(S1, D2)	45.00000	0.000000
MERCI_SD(S2, D1)	35.00000	0.000000
MERCI_SD(S2, D2)	0.000000	7.000000
COSTI_DD(D1, DD1)	3.000000	0.000000
COSTI_DD(D1, DD2)	5.000000	0.000000
COSTI_DD(D2, DD1)	6.000000	0.000000
COSTI_DD(D2, DD2)	4.000000	0.000000
MERCI_DD(D1, DD1)	35.00000	0.000000
MERCI_DD(D1, DD2)	0.000000	3.000000
MERCI_DD(D2, DD1)	0.000000	1.000000
MERCI_DD(D2, DD2)	45.00000	0.000000

Global optimal solution found.



Esempi di Problemi di Trasporto

Esercizio 1

Un problema di trasporto

Una ditta di trasporto deve trasferire delle casse dai propri magazzini ai principali negozi. Il trasporto ha un costo al Kmiglio per singola cassa di 90 dollari.

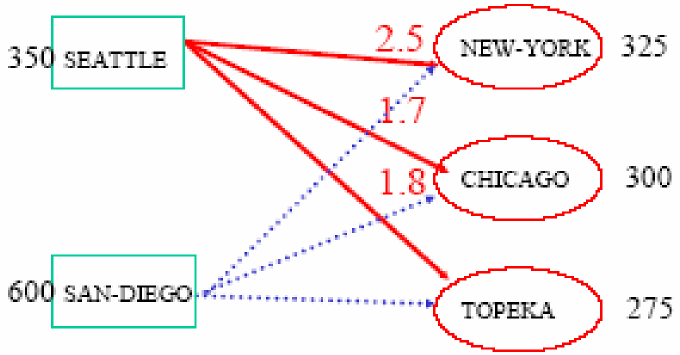
Le disponibilità di casse ai magazzini e le richieste ai negozi sono le seguenti:

Tabella

Magazzini	Disponibilità	Negozi	richieste
SEATTLE	350	NEW-YORK	325
SAN-DIEGO	600	CHICAGO	300
		TOPEKA	275

Distanze

	NEW-YORK	CHICAGO	TOPEKA
SEATTLE	2.5	1.7	1.8
SAN-DIEGO	2.5	1.8	1.4



x_{ij} = numero di casse trasportate dal magazzino i al venditore j = Costo del trasporto

c_{ij} = costo unitario del trasporto di una cassa da i a j

a_i = capacità del magazzino i $(i = 1,2)$

b_j = domanda del venditore j $(j = 1,3)$

$$f(x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}) = \sum_{i=1}^{2}\sum_{j=1}^{3} c_{ij}x_{ij} \quad \text{(costo totale del trasporto)}$$

$$\sum_{j=1}^{3} x_{ij} \le a_i \quad ; i = 1,2$$

$$\sum_{i=1}^{2} x_{ij} = b_j \quad ; j = 1,3$$

$$c_{ij} = Fd_{ij} \qquad (F = \text{costo per kmiglia} = 90\$, d_{ij} = \text{distanza magazzino - i negozio j})$$

Formulazione come problema di PL

$$Min \sum_{i=1}^{2} \sum_{j=1}^{3} Fd_{ij} x_{ij}$$

$$\sum_{j=1}^{3} x_{ij} \leq a_i \qquad ; i = 1,2$$

$$\sum_{i=1}^{2} x_{ij} = b_j \qquad ; j = 1,3$$

Condizione affinchè il problema ammetta soluzione :

$$\sum_{i=1}^{2} a_i \geq \sum_{j=1}^{3} b_i$$

L'offerta totale sia non inferiore alla domanda totale

Esercizio 1: Risolvere il problema con Lingo e verificare graficamente la soluzione ottenuta

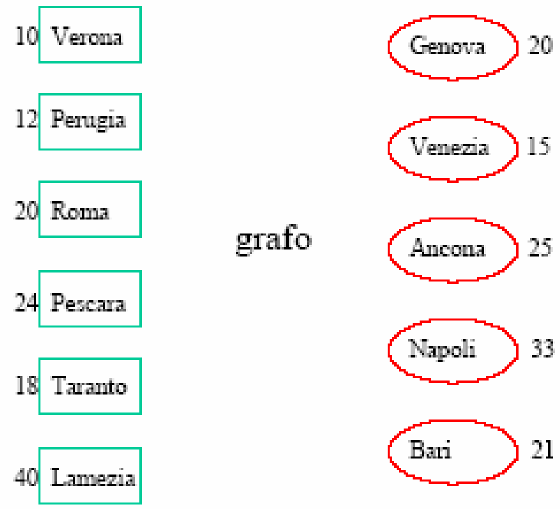
Esercizio 2

Un problema di trasporto

Una ditta di trasporto deve trasferire container vuoti dai propri magazzini ai principali porti nazionali
Le disponibilità di container vuoti ai magazzini e le richieste ai porti sono le seguenti:

Tabella

Magazzini	Disponi bilita	Porti	richieste
Verona	10	Genova	20
Perugia	12	Venezia	15
Roma	20	Ancona	25
Pescara	24	Napoli	33
Taranto	18	Bari	21
Lamezia	40		



Si assuma che il costo unitario a Km per 1 container sia di 0,5 Euro